

UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR



PROYECTO FIN DE CARRERA

**APLICACIÓN ANDROID PARA LA ENSEÑANZA DE
TEMAS DE CIRCUITOS INTEGRADOS BÁSICOS MOS**

Ana González Fernández

Enero 2015

APLICACIÓN ANDROID PARA LA ENSEÑANZA DE TEMAS DE CIRCUITOS INTEGRADOS BÁSICOS MOS

AUTOR: Ana González Fernández

TUTOR: Fernando Barbero Díaz

PONENTE: Eduardo Boemo Scalvinoni

Digital System Laboratory

Dpto. Tecnología Electrónica y de Comunicaciones

Escuela Politécnica Superior

Universidad Autónoma de Madrid

Enero 2015

Resumen

Este proyecto consiste en diseñar una aplicación gratuita para dispositivos Android, dónde se desarrollará un tutorial sobre circuitos integrados básicos MOS, una serie de ejercicios y un test. Los problemas han sido elegidos de la Guía N°1 de la asignatura Dispositivos Integrados Especializados, que se imparte en la Escuela Politécnica Superior de la Universidad Autónoma de Madrid.

El principal objetivo de la aplicación es que los alumnos entiendan el funcionamiento de las puertas básicas CMOS y ofrecer a los alumnos la posibilidad de realizar ejercicios de circuitos MOS desde cualquier dispositivo Android. Además la aplicación permite corregir los ejercicios automáticamente cuando el usuario lo desee y compartir las soluciones con otros usuarios.

La aplicación se ha desarrollado en inglés y se encuentra publicada en Google Play.

Palabras clave

Circuitos MOS, transistor, Android, móvil, tableta.

Abstract

This project consists on designing free application for android devices, which will develop a basic tutorial upon basic integrated circuits MOS and wide range of exercises. Those exercises have been picked up from the problem's guide number 1 of the subject integrated specialized devices offered in the Superior School of Polytechnics from Autonoma University of Madrid.

The main aim of this app is to understand how the basic CMOS door work and to offer the students the possibility of completing exercises concerning the MOS circuits through any Android device. In addition, the app permits the automatically correction of the exercises whenever the user needs it and the sharing of their results.

The app has been developed in English and has been published in Google Play

Key words

MOS Circuits, transistor, Mobile, Tablet

Agradecimientos

En primer lugar quiero agradecer a mis padres, Ángel y Mar, y a mis hermanos, por todo el apoyo que me han dado, por sus consejos y por que sin ellos no hubiese podido llegar hasta aquí.

También me gustaría agradecerle a Eduardo Boemo por haberme dado la oportunidad de realizado este proyecto.

Por último a los compañeros y amigos con los que he compartido tantos momentos durante la carrera, en especial a Marta, Joaquín, Fernando y Sergio.

INDICE DE CONTENIDOS

1 Introducción.....	1
1.1 Motivación.....	1
1.2 Objetivos.....	2
1.3 Organización de la memoria.....	6
2 Estado del arte	7
2.1 Evolución de los dispositivos móviles	7
2.2 Historia y arquitectura de Android	9
2.3 Aplicaciones similares.....	12
3 Diseño.....	15
3.1 Propósito del proyecto	15
3.2 Diagrama de estados del sistema	16
3.3 Requisitos	17
3.3.1 Sistema Operativo	17
3.3.2 Tipos de dispositivos	21
3.3.3 Tipos de pantalla.....	21
3.3.4 Idioma.....	24
3.3.5 Resumen de requisitos	25
3.4 Limitaciones	25
3.5 Módulos de la aplicación.....	27
3.5.1 Tutorial	27
3.5.2 Ejercicios	27
3.5.3 Guardar	36
3.5.4 Corregir.....	36
3.5.5 Mostrar solución	36
3.5.6 Resetear	36
3.5.7 Cargar y borrar ejercicios	37
3.5.8 Enviar ejercicios	37
3.5.9 Test	38
3.5.10 Manual de ayuda.....	38
3.5.11 Botón Menú de Android.....	39

4 Desarrollo	41
4.1 Primeros pasos.....	41
4.2 Herramientas utilizadas	41
4.3 Tipos de archivo	42
4.3.1 Activity	42
4.3.2 Views	44
4.3.3 Manifest	44
4.4 Esquema general.....	45
4.5 Menús de la aplicación	46
4.6 Tutorial	48
4.7 About	48
4.8 Diseño y desarrollo de ejercicios.....	49
4.8.1 Vistas	49
4.8.2 Actividad	51
4.9 Ejercicio 16.....	56
4.10 Preferencias	56
4.11 Cargar o eliminar soluciones guardadas.....	57
4.12 Enviar soluciones.....	58
4.13 Test	59
4.14 Adaptar a los diferentes tipos de pantalla.....	60
4.15 Diagrama y descripción de clases.....	61
4.16 Descripción métodos principales.....	63
5 Integración, pruebas y resultados	65
5.1 Pasos previos a la publicación en Google Play Store.....	65
5.1.1 Firmar una Aplicación	65
5.1.2 Alineamiento del “.apk”	68
5.2 Publicación en Google Play Store	69
5.2.1 Cuenta de desarrollador	69
5.2.2 Subir una aplicación	69
5.2.3 Actualizar una aplicación	70
5.3 Estadísticas proporcionadas por Google Play Store	71
5.3.1 Instalaciones totales de la aplicación.....	71
5.3.2 Instalaciones según versión de Android	71
5.3.3 Instalaciones totales por países.....	72

6 Conclusiones y trabajo futuro	73
6.1 Conclusiones.....	73
6.2 Encuesta.....	74
6.3 Trabajo futuro	78
Referencias	81
Glosario	83
Anexos.....	I

INDICE DE FIGURAS

Figura 1: Guía de problemas número 1	2
Figura 2: Icono Sequential Circuits	4
Figura 3: Icono Combinational Circuits	4
Figura 4: Icono Karn Map	5
Figura 5: Evolución de los dispositivos inteligentes	8
Figura 6: Arquitectura interna de Android	11
Figura 7: Diagrama de estados del sistema	16
Figura 8: Evolución cuota de mercado móviles en España	18
Figura 9: Evolución cuota de mercado tabletas en el mundo	18
Figura 10: Relación entre móviles y tabletas distribuidos.....	19
Figura 11: Cuota de dispositivos Android por versión (3 de Noviembre de 2014).....	21
Figura 12: Clasificación de las pantallas Android según tamaño y densidad	22
Figura 13: Distribución de dispositivos según tamaño de pantalla	22
Figura 14: Distribución de dispositivos según densidad de pantalla.....	23
Figura 15: Ejercicio con 4 entradas	26
Figura 16: Ejercicio de entradas y 12 transistores	26
Figura 17: Botón “Back”	42
Figura 18: Ciclo de vida de una actividad.	43
Figura 19: AndroidManifest.xml.....	44
Figura 20: Esquema general de la aplicación	45
Figura 21: Menú principal	46
Figura 22: Menú ejercicios	46
Figura 23: Menú ayuda.....	47
Figura 24: Menú accesible desde el botón “menú” de Android	47
Figura 25: Aviso tutorial	48
Figura 26: Pantalla “Acerca de”	49
Figura 27: Vistas ejercicios 3 y 4 entradas	51
Figura 28: Diálogo Guardar ejercicio	53
Figura 29: Sobrescribir ejercicio	53
Figura 30: Corregir ejercicio	54
Figura 31: Mostrar solución del ejercicio.....	55

Figura 32: Aviso confirmación borrar ejercicio	55
Figura 33: Ejercicio 16	56
Figura 34: Enviar solución por correo	59
Figura 35: Puntuación test y preguntas erróneas	60
Figura 36: Diagrama de clases.....	61
Figura 37. Firmar aplicación: Use the Export Wizard.....	65
Figura 38. Firmar Aplicación: seleccionar proyecto a exportar	66
Figura 39. Firmar Aplicación: keystore.....	67
Figura 40. Firmar Aplicación: Key creation.....	67
Figura 41. Firmar Aplicación: destino final proyecto.apk.....	68
Figura 42: Alineamiento: zipalign.....	69
Figura 43: Descargas totales de la aplicación.....	71
Figura 44: Instalaciones actuales a 09/12/2014.....	71
Figura 45: Descargas totales según versión de Android.....	72
Figura 46: Instalaciones totales por país	72
Figura 47: Encuesta Organización general	75
Figura 48: Encuesta Aspecto de los gráficos.....	75
Figura 49: Encuesta Facilidad de uso	76
Figura 50: Fiabilidad y estabilidad	76
Figura 51: Encuesta Consumo de batería	76
Figura 52: Encuesta Velocidad.....	77
Figura 53: Encuesta Ordenación de los ejercicios.....	77
Figura 54: Ingresos publicidad/idioma	79

INDICE DE TABLAS

Tabla 1: Versiones Android.....	10
Tabla 2: Comparación aplicaciones de Android	13
Tabla 3: Distribución de dispositivos según sistema operativo.....	19
Tabla 4: Tabla de versiones activas de Android a 3 de Noviembre de 2014	20
Tabla 5: Distribución de dispositivos según tamaño de pantalla	22
Tabla 6: Distribución de dispositivos según densidad de pantalla	23
Tabla 7: Distribución de dispositivos según tamaño y densidad.....	24
Tabla 8: Ejercicio 1.....	28
Tabla 9: Ejercicio 2.....	28
Tabla 10: Ejercicio 3.....	29
Tabla 11: Ejercicio 4.....	29
Tabla 12: Ejercicio 5.....	30
Tabla 13: Ejercicio 6.....	30
Tabla 14: Ejercicio 7.....	31
Tabla 15: Ejercicio 8.....	31
Tabla 16: Ejercicio 9.....	32
Tabla 17: Ejercicio 10.....	32
Tabla 18: Ejercicio 11.....	33
Tabla 19: Ejercicio 12.....	33
Tabla 20: Ejercicio 13.....	34
Tabla 21: Ejercicio 14.....	34
Tabla 22: Ejercicio 15.....	35
Tabla 23: Ejercicio 16.....	35
Tabla 24: Descripción métodos principales.....	63
Tabla 25: Número de descargas según versión de Android	71
Tabla 26: Encuesta.....	74

1 Introducción

1.1 Motivación

En muy pocos años los teléfonos móviles han llegado a convertirse en un dispositivo esencial en nuestras vidas, se han desarrollado dispositivos con características cercanas a las de algunos ordenadores personales. Hoy en día estos dispositivos se han extendido tanto, que prácticamente todo el mundo dispone de uno de ellos.

Este proyecto surge con la idea de aprovechar las ventajas que ofrecen estos dispositivos inteligentes en los estudios universitarios.

Actualmente las guías de problemas de la asignatura Dispositivos Integrados Especializados de la Escuela Politécnica Superior de la Universidad Autónoma de Madrid se entregan en papel y se publican en formato PDF, cuyo resultado final sigue siendo una hoja de papel. Además estas guías no traen las soluciones parciales, ni tampoco corrigen todos los ejercicios.

La idea es desarrollar una aplicación para la plataforma Android, dónde los alumnos puedan encontrar las guías de problemas con una herramienta de corrección, tutoriales sobre temas concretos y la posibilidad de enviar los resultados a otros compañeros o al profesor.

Adicionalmente, entre las ventajas que ofrecen estos dispositivos podemos destacar:

- Comodidad de uso.
- Eliminación del uso del papel.
- Utilización sin necesidad de internet.
- Acceso rápido a la aplicación a cualquier hora y en cualquier lugar del mundo que tenga acceso al mercado de las aplicaciones móviles.
- Posibilidad de modificación y redistribución automática a través de Google Play.
- Mayor protección intelectual frente a plagios.

1.2 Objetivos

El objetivo principal de este proyecto es desarrollar una aplicación Android gratuita destinada a los alumnos de la asignatura Dispositivos Integrados Especializados de la EPS de la Universidad Autónoma de Madrid, que ofrezca 16 problemas, un tutorial y un test, dedicados a los circuitos integrados básicos MOS.

Los problemas se centrarán en el análisis de circuitos MOS. La aplicación ofrecerá la posibilidad de corregir los problemas y comprobar sus soluciones. Además los ejercicios se podrán guardar, cargar y enviar por correo para una mayor interacción entre alumnos y con el profesor.

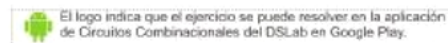
El tutorial y los ejercicios tratan de ayudar al alumno a comprender los siguientes conceptos:

- Puertas básicas CMOS
- Llave tri-state
- Cómo se hace un mux con transistores
- Cómo aumentar el driving

Esta aplicación debe complementar la guía de problemas de la asignatura DIE, en la cual se indicará que ejercicios están disponibles en el teléfono:

DISPOSITIVOS INTEGRADOS ESPECIALIZADOS
Escuela Politécnica Superior, Universidad Autónoma de Madrid
Guía de Problemas N° 1: Puertas Básicas CMOS

Objetivos: Entender a) cómo son las puertas básicas CMOS; b) cómo es una llave *tri-state*; c) cómo se hace un *mux* con transistores; d) cómo se consigue aumentar el *driving*; e) qué funciones y puertas lógicas se adaptan mejor a la tecnología CMOS



1. Utilizando la tabla de la figura (donde P_i significa transistor P izquierdo, etc.) indique el estado de cada transistor (conducción-corte, etc.) y deduzca el nivel lógico de la salida O. ¿A qué puerta corresponde?

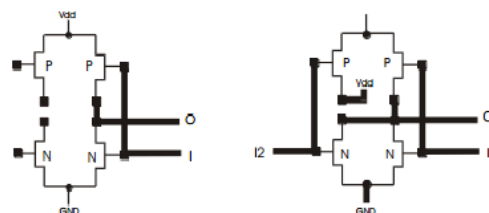


Figura 1: Guía de problemas número 1

La aplicación debe seguir estrictamente el estilo de dos aplicaciones anteriores realizadas en el laboratorio para la asignatura Circuitos Electrónicos Digitales del primer curso de Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación de la Escuela Politécnica Superior de la UAM. Estas restricciones incluyen aspectos como:

- Tamaño de la letra
- Colores
- Formato del menú inicial
- Icono
- Resolución de gráficos
- Relación de aspecto de gráficos
- Formato de botones
- Menú de ayuda
- Página “about”
- Funcionamiento del tutorial

Desde el punto de vista educativo el objetivo del proyecto es adquirir los siguientes conocimientos:

- Programación orientada a objetos: JAVA.
- Lenguaje XML.
- Conocimiento de la plataforma Android (AVD, SDK).
- Publicar aplicaciones en Google Play Store.

La aplicación será válida tanto para móviles como para tabletas y será ofrecida a los alumnos a través de la plataforma de Google Play, en la que podrán descargarla de manera gratuita, valorarla, opinar y proponer mejoras. No será necesaria la conexión a Internet para utilizar los contenidos de la aplicación.

Los objetivos mencionados con anterioridad forman parte de los objetivos concretos de este proyecto, pero esta aplicación forma parte de proyecto mayor. En el laboratorio de la UAM ya se han desarrollado otras tres aplicaciones con el propósito de cambiar la forma de realizar las guías de ejercicios y facilitar a los usuarios el estudio de otras asignaturas desde su dispositivo Android.

A continuación vamos a explicar las diferentes aplicaciones educativas desarrolladas por el laboratorio:

Sequential Circuits



Figura 2: Icono Sequential Circuits

Esta aplicación fue publicada en diciembre de 2013, y se caracteriza por ofrecer una serie de ejercicios de electrónica secuencial (máquinas de estados), incluye consejos para ayudar a resolver estos problemas, un tutorial centrado en máquinas de estados tipo Moore y un manual con el manejo básico de la aplicación.

Los contenidos de esta aplicación abarcan temas como: Mealy, Moore, Flip Flops, memoria ROM, puertas lógicas, detector de secuencias, aritmética serie, solapamiento y control de motores.

Un año después de la publicación de la primera versión, la aplicación cuenta con 1.800 descargas y una nota media de 4,14 sobre 5 en las valoraciones.

Combinational Circuits

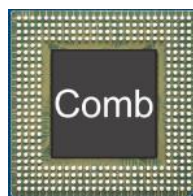


Figura 3: Icono Combinational Circuits

Está centrada en ejercicios de electrónica combinacional, al igual que la anterior incluye consejos para ayudar a resolver los problemas. También dispone de un manual que explica cómo utilizar la aplicación y un tutorial sobre circuitos combinacionales.

Esta aplicación se centra en: memoria ROM, multiplexores, puertas lógicas, PALs, BCD 7-segmentos, Codificador y decodificador, código Gray y Karnaugh.

Su primera versión se publicó el 16 de diciembre de 2013, en la actualidad se han publicado un total de 7 versiones. La aplicación se ha descargado aproximadamente 3.200 veces y la nota media de sus valoraciones es un 4 sobre 5.

Karn Map



Figura 4: Icono Karn Map

Ofrece una serie de herramientas para la simplificación de funciones lógicas. Si la función tiene de 3 a 5 variables tiene la opción de la simplificación automática y gráfica con mapas de Veitch-Karnaugh. De 3 a 7 variables simplifica funciones introducidas por expresión lógica, marcando una tabla de verdad, eligiendo minitérminos o maxitérminos. Además dibuja el esquemático de dicha función.

Los contenidos tratados son: simplificación por mapas Veitch-Karnaugh, simplificación de hasta 7 variables, tabla de verdad, sumatorio de minitérminos, producto de maxitérminos, expresiones lógicas, representación esquemática, conversor, binario, octal y hexadecimal.

Esta aplicación es la más reciente y a pesar de llevar publicada en Play Store apenas dos meses, ya se ha instalado en más de 1.000 dispositivos. Y al igual que las anteriores la media obtenida de las valoraciones realizadas por los usuarios es muy alta, en este caso 4,78.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- Capítulo 1: Introducción
- Capítulo 2: Estado del arte
- Capítulo 3: Diseño del proyecto
- Capítulo 4: Desarrollo del proyecto
- Capítulo 5: Integración, pruebas y resultados
- Capítulo 6: Conclusiones y trabajo futuro

2 Estado del arte

En este apartado se va a realizar un pequeño estudio sobre la evolución de los dispositivos móviles y los principales sistemas operativos activos, en especial Android ya que es el elegido para la realización del proyecto. Además de un estudio del mercado de las aplicaciones educativas relacionadas con los circuitos integrados básicos MOS.

2.1 Evolución de los dispositivos móviles

La invención del teléfono móvil se remonta a los inicios de la Segunda Guerra Mundial, donde se hizo necesaria la comunicación a distancia, es por eso que Motorola creó un equipo llamado *Handie Talkie H12-16*, basado en la transmisión mediante ondas de radio. Esta tecnología se usó para crear gran variedad de aparatos de radio y comunicación a distancia, aunque no pueden ser considerados como teléfonos móviles, estos dispositivos supusieron el comienzo de la evolución hacia los dispositivos que conocemos en la actualidad.

La primera generación de teléfonos móviles surgió en 1973, los móviles con esta tecnología funcionaban de manera analógica, es decir la transmisión y recepción de datos se apoyaba sobre un conjunto de ondas de radio que cambiaban de modo continuo. Lo que permitía a cualquier persona escuchar llamadas ajenas con un simple sintonizador de radio. La denominada segunda generación marca el paso de la telefonía analógica a la digital, que permitió la mejora del manejo de llamadas y la integración de otros servicios adicionales al de la voz, entre los que destacan los SMS.

En cuanto a la tercera generación surge de la necesidad de aumentar la capacidad de transmisión de datos, para poder ofrecer otros servicios como la conexión a internet desde el móvil, la videoconferencia, la televisión y la descarga de archivos. Es entonces cuando surgen los primeros dispositivos móviles inteligentes o Smartphones.

En 1993 fue lanzado al mercado el “Simon” de IBM, considerado el primer Smartphone. El equipo funcionaba como un teléfono móvil, un asistente personal digital e incluso como un máquina de fax. Tenía pantalla táctil y el texto se ingresaba mediante un teclado QWERTY.

Posteriormente Nokia y Palm, lanzaron más dispositivos de este tipo, pero no triunfaron en el mercado a pesar de sus avances tecnológicos.

En 1999 sale al mercado la primera BlackBerry, que permitía acceder al correo electrónico y navegar por internet. Estos dispositivos tuvieron una gran repercusión en el mercado.

En 2007 Apple lanza el iPhone, lo que supone el verdadero estallido de los Smartphones. Se vendieron millones de unidades de este dispositivo, que sin duda ofrecía la mejor conexión a Internet desde un teléfono móvil hasta la fecha. También en este año Google presenta el sistema operativo Android, que actualmente es el sistema operativo móvil más utilizado en el mundo.



Figura 5: Evolución de los dispositivos inteligentes

Desde entonces los dispositivos móviles inteligentes se han expandido muy rápido, hoy en día prácticamente todo el mundo tiene uno. A la vez se han desarrollado también las tabletas, que son dispositivos de características similares a los Smartphone pero con una pantalla más grande. Las tabletas comenzaron a expandirse en 2010 con el lanzamiento del iPad.

2.2 Historia y arquitectura de Android

Android es un sistema operativo basado en el kernel de Linux diseñado principalmente para dispositivos móviles con pantalla táctil, inicialmente desarrollado por Android Inc., empresa que fue comprada por Google en 2005.

En 2007 se fundó la Open Handset Alliance (OHA), una alianza comercial entre compañías telefónicas o de telecomunicaciones, que se dedica a desarrollar estándares abiertos para dispositivos móviles, en ese mismo año se presentaba el sistema operativo Android.

El primer móvil con este sistema operativo se vendió en octubre del 2008, el Smartphone HTC Dream.

Android permite el control de los dispositivos por medio de bibliotecas desarrolladas por Google mediante el lenguaje de programación Java. Es un sistema operativo de código abierto lo que quiere decir que cualquier desarrollador puede crear aplicaciones en lenguaje “C” u otro lenguaje y compilarlas a código nativo de ARM (API de Android). Además permite que cualquier desarrollador pueda acceder al código fuente para modificarlo y mejorarlo. Por otro lado, los costes para lanzar una aplicación en Android son muy bajos, ya que sólo hay que pagar una cuota inicial al darse de alta como desarrollador.

Gracias a la facilidad para publicar aplicaciones, Android cuenta con una tienda con más de un millón de aplicaciones, conocida como Google Play. Es un catálogo de aplicaciones gratuitas o de pago, que pueden ser descargadas e instaladas en el dispositivo a través de la aplicación móvil Play Store. Lo que ha hecho que Android sea el SO para Smartphones más usado en la actualidad.

Características

Algunas de las principales características de este sistema operativo son:

- Diseño del dispositivo
- Conectividad
- Navegador Web
- Soporte multimedia
- Soporte para streaming
- Entorno de desarrollo

- Multi-táctil
- Multitarea
- Tethering

Versiones

A continuación se muestra una tabla con las diferentes actualizaciones, que han ido corrigiendo errores o añadiendo nuevas funcionalidades. Desde la publicación de la primera versión en 2008 se han desarrollado veinte versiones con distinta interfaz de programación de aplicaciones (API).

Nombre de la versión	Número de versión	API
Beta	-	-
Apple Pie	1.0	1
Banana Bread	1.1	2
Cupcake	1.5	3
Donut	1.6	4
Eclair	2.0	5
Eclair 0.1	2.0.1	6
Eclair MR1	2.1	7
Froyo	2.2	8
Gingerbread	2.3	9
Gingerbread MR1	2.3.3	10
Honeycomb	3.0	11
Honeycomb MR1	3.1	12
Honeycomb MR2	3.2	13
Ice Cream Sandwich	4.0	14
Ice Cream Sandwich MR1	4.0.3	15
Jelly Bean	4.1	16
Jelly Bean MR1	4.2	17
Jelly Bean MR2	4.3	18
Kitkat	4.4	19
Lollipop	5.0	21

Tabla 1: Versiones Android

Arquitectura

La arquitectura interna de la plataforma Android, está formada básicamente por cuatro capas, todas ellas de software libre. Cada una de estas capas utiliza servicios ofrecidos por las capas anteriores, y ofrece a su vez sus propios servicios a las capas de niveles superiores.

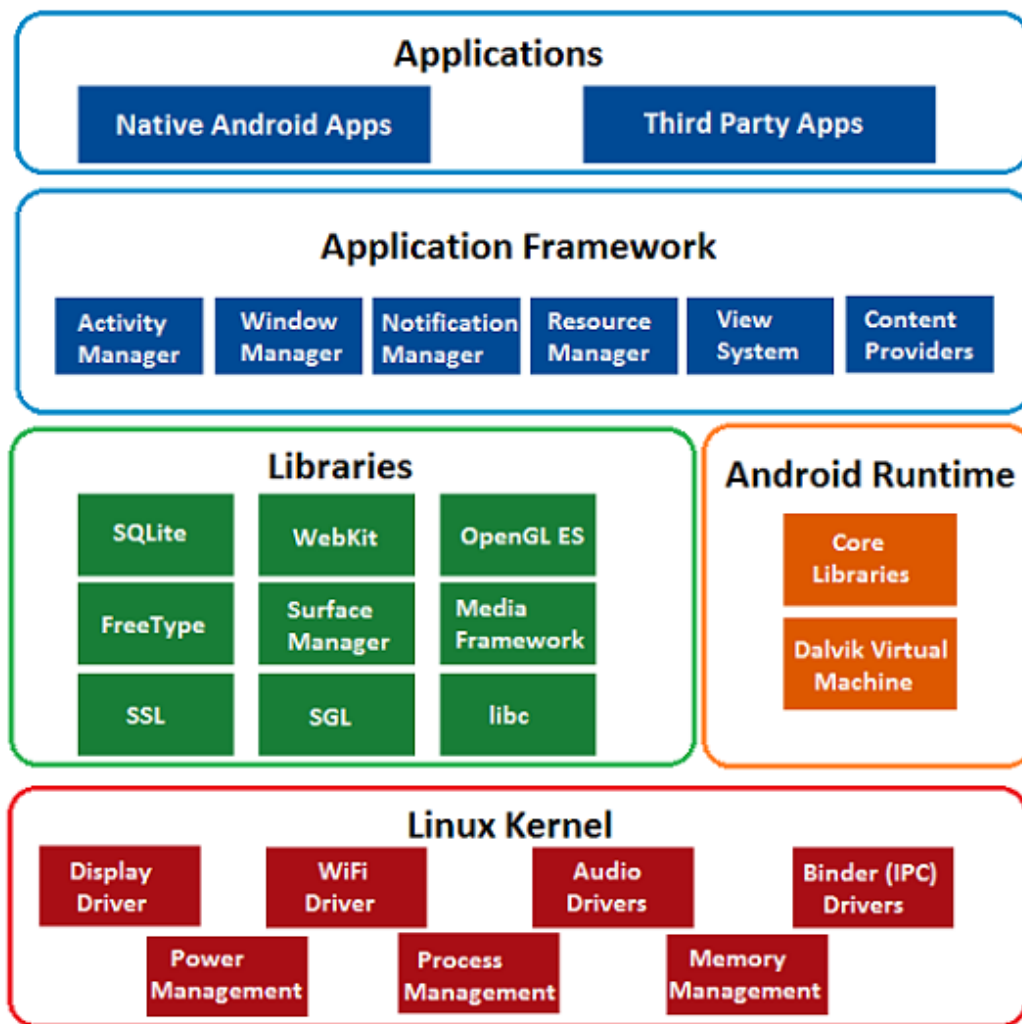


Figura 6: Arquitectura interna de Android

Aplicaciones: Este nivel contiene todas las aplicaciones, tanto las incluidas por defecto en Android como todas aquellas que el usuario vaya añadiendo posteriormente, ya sean de terceras empresas o de su propio desarrollo. Todas las aplicaciones han de correr en la máquina virtual Dalvik para garantizar la seguridad del sistema. Todas ellas utilizan los servicios, las API y librerías de los niveles anteriores.

Entorno de aplicación: Aquí se encuentran las clases y servicios que utilizan las aplicaciones para realizar sus funciones. Estas clases utilizan las bibliotecas y la máquina virtual para trabajar. Esta capa ha sido diseñada para simplificar la reutilización de componentes, ya que las aplicaciones pueden publicar sus capacidades y otras pueden hacer uso de ellas.

Bibliotecas: Incluye un conjunto de librerías escritas en C/C++, que proporcionan a Android la mayor parte de sus capacidades más características. Junto al núcleo basado en Linux estas librerías constituyen el corazón de Android.

Runtime de Android: Al mismo nivel que las bibliotecas de Android se encuentra el entorno de ejecución. Que incluye las bibliotecas esenciales de Android, que a su vez contienen las librerías propias de Java y otras específicas de Android, y la máquina virtual Dalvik. Algunas de las características de esta máquina que facilitan la optimización de recursos son: ejecuta ficheros ejecutables “.dex” –formato optimizado para ahorrar memoria, más compacto que los ficheros “.class”. Además Dalvik se basa en registros para guardar los datos y no en pila, por lo que ejecuta las instrucciones más rápido.

Núcleo de Linux: Android utiliza el núcleo de Linux 2.6 como una capa de abstracción para el hardware disponible en los dispositivos móviles. Esta capa contiene los drivers necesarios para que cualquier componente hardware pueda ser utilizado mediante las llamadas correspondientes. Siempre que un fabricante incluye un nuevo elemento hardware lo primero que debe hacer, para que pueda ser utilizado, es crear las librerías de control o drivers necesarios dentro del kernel de Linux en el propio Android.

2.3 Aplicaciones similares

Antes de desarrollar una aplicación es importante estudiar si el mercado ofrece algo similar. Para ello vamos a centrarnos en los principales sistemas operativos, que como ya hemos visto son: Android, iOS y Windows Phone. Cada uno de ellos tiene su propia tienda de aplicaciones, Google Play Store es la de Android, App Store de iOS y Windows Phone Store la de Windows Phone.

En nuestro caso la principal función de la aplicación es ayudar a los alumnos de la universidad a afianzar los conceptos acerca de circuitos MOS. La aplicación destaca por la posibilidad de realizar ejercicios, corregirlos y comprobar su solución. Además dispone de un test que permite a los alumnos evaluar sus conocimientos.

Tras el estudio de aplicaciones educativas en los diferentes mercados, podemos destacar que en **iOS** y en **Windows Phone** no se han encontrado aplicaciones similares a la nuestra. En el caso de Windows Phone tiene sentido ya que es un sistema operativo que ha nacido recientemente; la comunidad de desarrolladores está creciendo e inicialmente están portando las aplicaciones más usadas de Android e iOS (Google apps, Dropbox,

Shazam...). Para el sistema operativo de Apple habría que excusarse en que es un sistema diseñado para usuarios “menos avanzados”, pese a que cuenta con un gran número de aplicaciones en la tienda, el diseño de aplicaciones educativas tan técnicas está más reducido.

Respecto a las aplicaciones encontradas en **Android** hemos realizado una tabla comparativa:


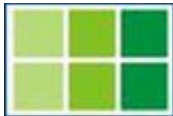



Nombre	CMOS Lite	Cmos4000	MOS ICs and Technology	Mosfet_calculati on	MOS Circuits
Logo					
Interfaz	Simple e intuitiva	Simple	Simple, recargada de anuncios	Difícil de usar	Simple e intuitiva
Velocidad	Buena	Normal	Un poco lenta debido a los anuncios	Normal	Buena
Idioma	Inglés	Alemán	Inglés	Inglés	Inglés
Usabilidad	Se realizan preguntas acerca de circuitos MOS y se puede comprobar la respuesta.	Ofrece las fichas técnicas de los diferentes circuitos integrados CMOS de la serie 4000	Aplicación destinada a estudiantes, cubre 114 temas de circuitos MOS en detalle. Es como un libro, sólo teoría.	Permite realizar cálculos simples de circuitos de transistores MOSFET.	Aplicación destinada a estudiantes, consta de un tutorial, ejercicios de práctica y un test.

Tabla 2: Comparación aplicaciones de Android

3 Diseño

En este apartado se explicarán algunas de las elecciones tomadas a la hora de diseñar la aplicación y se describirán las principales propiedades, requisitos y limitaciones del proyecto.

3.1 Propósito del proyecto

El principal objetivo de este proyecto es facilitar el estudio de los circuitos integrados básicos MOS a los alumnos de la asignatura Dispositivos Integrados Especializados de la Universidad Autónoma de Madrid. Aunque también será válido para cualquier usuario que esté interesado en la temática que se ofrece.

Para ello se ha desarrollado una aplicación para móviles y tabletas, ya que es evidente que están muy presentes entre los estudiantes. La idea es mejorar así el formato de las guías de problemas de la asignatura, ya que el alumno puede realizar los problemas de las guías y corregirlos de manera interactiva, sin necesidad de utilizar bolígrafo y papel.

La aplicación ofrece un tutorial sobre los transistores MOS, y una guía de 16 ejercicios de circuitos MOS. El alumno podrá corregir los ejercicios y comprobar la solución, también podrá guardar y cargar las soluciones, y enviarlas por correo. Dispondrá de un manual de usuario de la aplicación. Además podrá hacer un test de 12 preguntas con 4 alternativas, para comprobar sus conocimientos.

Como hemos podido comprobar al revisar el mercado de aplicaciones educativas relacionadas con este tema, no hay ninguna que haga factible realizar ejercicios de circuitos MOS, con la posibilidad de corregirlos.

Como el único fin de este proyecto es educativo, la aplicación es gratuita.

3.2 Diagrama de estados del sistema

Un objetivo fundamental del proyecto es la facilidad de manejo de la aplicación. Para ello se ha creado una interfaz con botones simples e intuitivos, que hacen que la aplicación sea muy fácil de utilizar.

La aplicación inicia con el menú principal, en el que se podrá elegir entre cinco opciones: Tutorial, Ejercicios, Enviar Ejercicios, Test o Ayuda. Dependiendo del módulo elegido se podrá acceder a las diversas funciones que posee el proyecto y que se explicarán más adelante en profundidad.

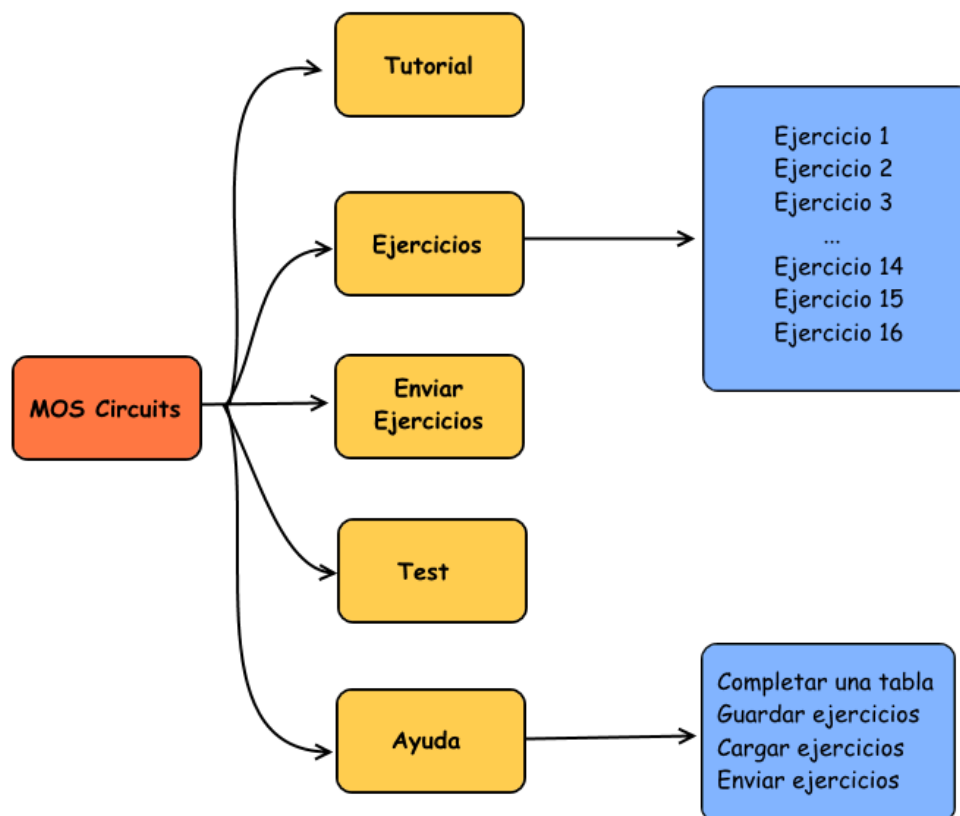


Figura 7: Diagrama de estados del sistema

Como podemos observar el diseño de la aplicación está pensado para que el usuario de la aplicación pueda navegar sin ninguna dificultad. Es un diseño en árbol, en el que a partir de un contenido general se accede a uno más específico.

3.3 Requisitos

3.3.1 Sistema Operativo

Hemos comparado los tres sistemas operativos con mayor repercusión (Android, iOS y Windows Phone) desde el punto de vista de un desarrollador. Entre ellos, el sistema operativo elegido para nuestra aplicación ha sido Android, las principales razones se explicarán a continuación.

Desde un primer momento uno de los principales objetivos del proyecto ha sido publicar la aplicación para que estuviese a disposición de los alumnos. Como ya hemos visto en el estado del arte (2.2 Principales sistemas operativos), los tres sistemas disponen de plataformas para publicar aplicaciones. Sin embargo Android es la más económica ya que para hacerse desarrollador y poder publicar aplicaciones en su tienda, habrá que realizar un único pago de 25 dólares. Mientras que las cuotas para publicar aplicaciones en iOS y Windows Phone son anuales y bastante superiores.

Además Android es una plataforma libre y de código abierto, en la que se pueden descargar todas las herramientas necesarias de manera gratuita desde la página oficial de desarrolladores:

<http://developer.android.com/sdk/index.html>

Estas herramientas son multiplataforma, es decir están disponibles para los principales sistemas operativos de PC. En el caso de iOS sólo es posible desarrollar con dispositivos de la marca Apple, y en el caso de Windows Phone con dispositivos que tengan como mínimo Windows 7. Una vez desarrollada la aplicación, en Android se puede instalar en cualquier dispositivo compatible para probarla, en iOS con una cuenta de desarrollador gratuita sólo se puede probar las aplicaciones en el emulador y en el caso de Windows Phone habría que crear primero la cuenta de desarrollador para poder registrar el dispositivo en el que realizar las pruebas.

Por otro lado al ser Android una plataforma de código abierto, cualquier desarrollador tiene acceso al código por lo que existen continuas mejoras. Cuenta con un gran número de desarrolladores que proporcionan información y soluciones a cualquier problema que pueda surgir, facilitando así el desarrollo de aplicaciones.

Por último y más importante, hay que destacar que Android está cada día más presente en los dispositivos móviles. En los últimos 4 años ha ido creciendo y ocupando el primer puesto en el mercado, tanto en España como en el resto del mundo.

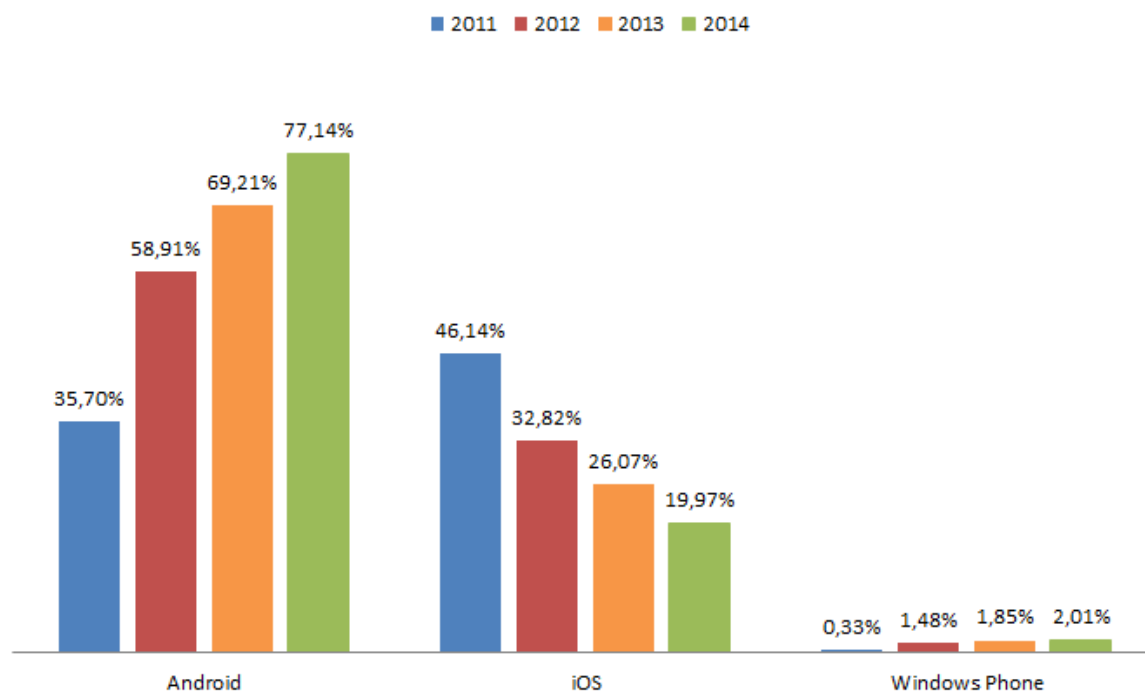


Figura 8: Evolución cuota de mercado móviles en España

Como podemos comprobar el en siguiente gráfico no ocurre lo mismo con las tabletas, ya que iOS es el líder absoluto con el iPad.

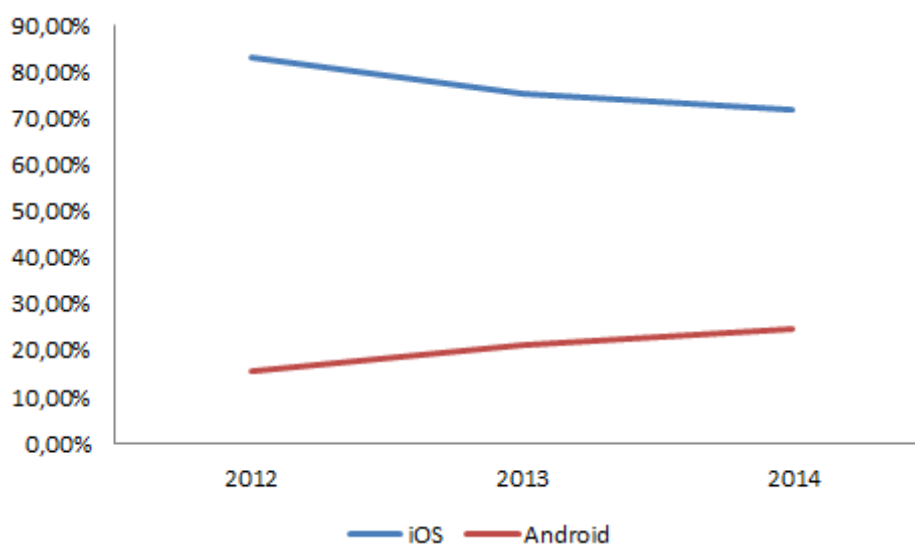


Figura 9: Evolución cuota de mercado tabletas en el mundo

Aun así nos decantamos por el sistema operativo Android por dos motivos, el primero es que cómo es lógico hay muchos más móviles que tabletas, como podemos comprobar en el siguiente gráfico:

Relación entre móviles y tabletas en 2014

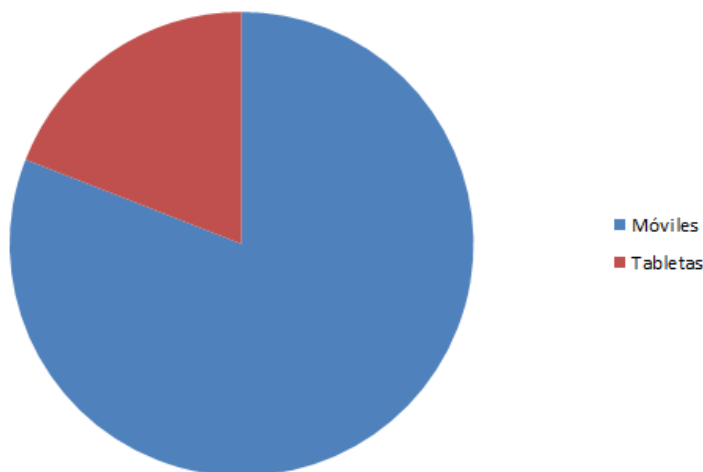


Figura 10: Relación entre móviles y tabletas distribuidos

Y el segundo es que al comparar los tres sistemas operativos más extendidos en términos de dispositivos totales (móviles y tabletas), es Android el que tiene una cuota de mercado mucho mayor con un 84.7% de los dispositivos totales, frente al 11,7% de iOS y el 2,5% que tiene Windows Phone.

	Android	iOS	Windows Phone	Others
2011	36.1%	18.3%	1.2%	44.4%
2012	69.3%	16.6%	3.1%	11%
2013	79.6%	13.0%	3.4%	4%
2014	84.7%	11.7%	2.5%	1.2%

Tabla 3: Distribución de dispositivos según sistema operativo

De manera que eligiendo Android como sistema operativo para el proyecto nos aseguramos de que la aplicación estará disponible para el mayor número de personas posible.

Versiones válidas

Gracias a la plataforma de desarrollo de Android, la elección del rango de versiones para las que estará disponible la aplicación se realiza de manera muy sencilla, en el archivo `AndroidManifest.xml` se puede indicar la versión mínima (`android: minSdkVersion="8"`) y la máxima (`android: targetSdkVersion="19"`) que soportará la aplicación.

Es importante mencionar que funciones creadas para versiones nuevas no serán soportadas por versiones antiguas. Pero cualquier versión nueva soportará funciones creadas para una versión antigua.

Para establecer el rango de versiones hemos realizado un pequeño estudio, teniendo en cuenta los datos proporcionados por Android relativos al número de dispositivos que corren con las diferentes versiones.

Versión	Codename	API	Distribution
2.2	Froyo	8	0.6%
2.3.3- 2.3.7	Gingerbread	10	9.8%
4.0.3- 4.0.4	Ice Cream Sandwich	15	8.5%
4.1.x	Jelly Bean	16	22.8%
4.2.x	Jelly Bean	17	20.8%
4.3	Jelly Bean	18	7.3%
4.4	KitKat	19	30.2%

Tabla 4: Tabla de versiones activas de Android a 3 de Noviembre de 2014

Todas las versiones con un porcentaje menor a 0.1 no se muestran en la tabla.

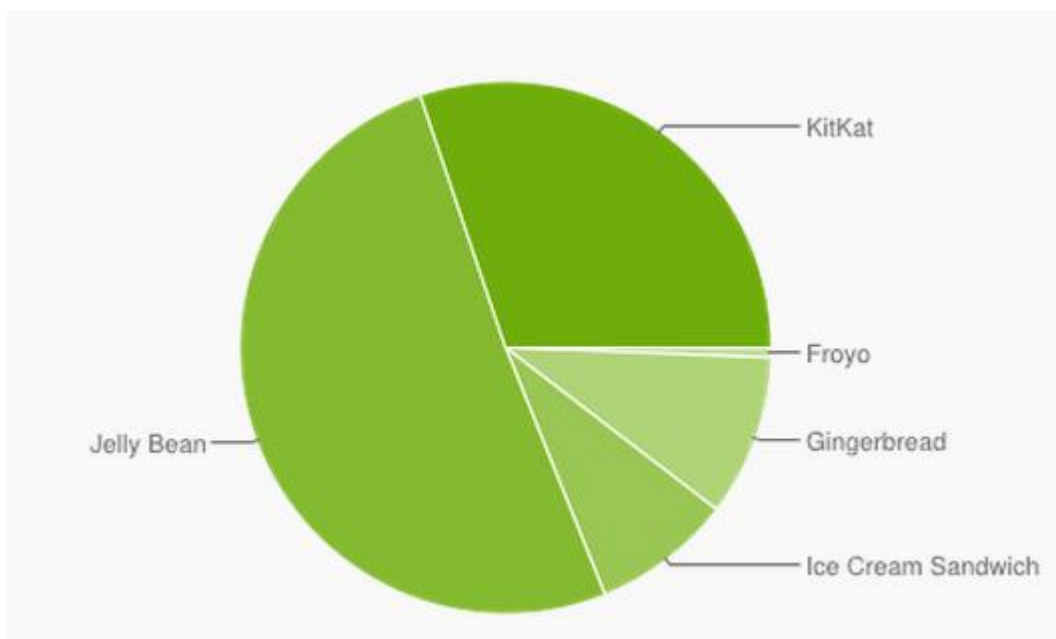


Figura 11: Cuota de dispositivos Android por versión (3 de Noviembre de 2014)

Teniendo en cuenta estos datos hemos establecido que la versión mínima sea la 2.2 Froyo y la máxima 4.4 KitKat. De manera que nuestra aplicación es compatible con casi el 100% de los dispositivos móviles existentes. A medida que vayan surgiendo nuevas versiones habrá que ir adaptando la aplicación para que las soporte.

3.3.2 Tipos de dispositivos

Nuestro principal objetivo es facilitar la realización de las guías de problemas a los estudiantes, por lo que los dispositivos elegidos para los que desarrollar la aplicación son los móviles y las tabletas. No sólo por la presencia diaria entre los estudiantes sino también por la facilidad para transportarlos, lo que permite que el alumno pueda acceder a la aplicación desde cualquier sitio y a cualquier hora.

3.3.3 Tipos de pantalla

Cuando hablamos de tipos de pantalla tenemos que diferenciar entre el tamaño de la pantalla y la densidad. El tamaño de la pantalla es la medida real de la pantalla en diagonal, la distancia en pulgadas de punta a punta del cristal de la pantalla. Mientras que la densidad es una medida que cuantifica el número de píxeles en un área física de la pantalla, medidas en dpi (dots per inch). Cuanto mayor sea la densidad de las pantallas mayor será el detalle con el que se mostrará el contenido.

En la actualidad Android ofrece gran variedad de productos con diferentes tamaños y densidades. Android clasifica las pantallas según sus diferentes tamaños y densidades de la siguiente manera:

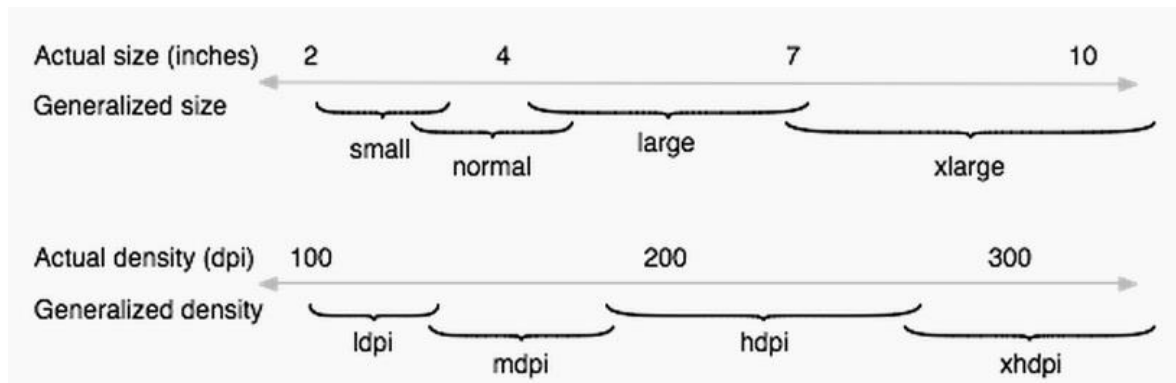


Figura 12: Clasificación de las pantallas Android según tamaño y densidad

A continuación se mostrarán la distribución de los dispositivos según su tamaño y densidad, estos datos son proporcionados por Android y están actualizados a noviembre de 2014.

Tamaño	Total distribuido
Small	5.8%
Normal	81.4%
Large	8.1%
Xlarge	4.7%

Tabla 5: Distribución de dispositivos según tamaño de pantalla

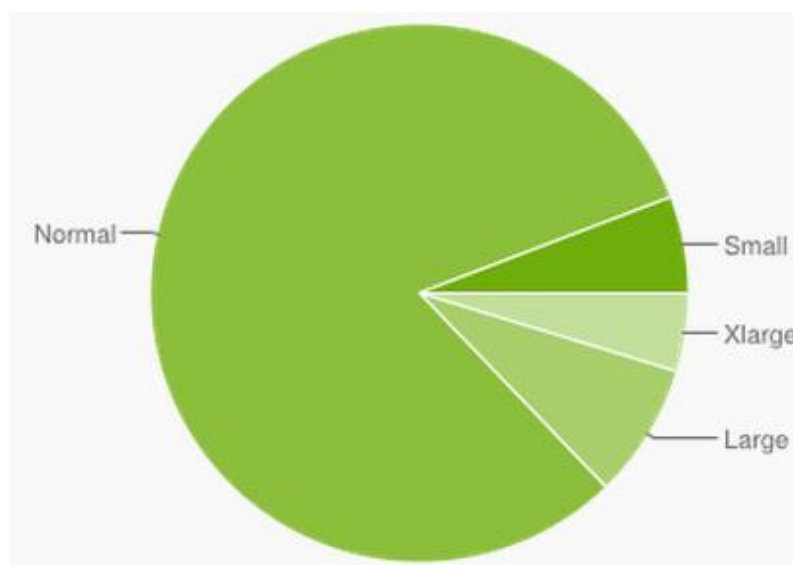


Figura 13: Distribución de dispositivos según tamaño de pantalla

Cómo podemos observar en los gráficos el tamaño predominante es el normal, aunque las pantallas grandes están cada vez más presentes. Hemos decidido ajustar la aplicación a los cuatro tipos de pantalla ya que nuestro objetivo es que sea compatible con el mayor número de dispositivos posible.

Densidad	Total distribuido
ldpi	6.3%
mdpi	18.3%
tvdpi	1.9%
hdpi	37.5%
xhdpi	20.0%
xxhdpi	16.0%

Tabla 6: Distribución de dispositivos según densidad de pantalla

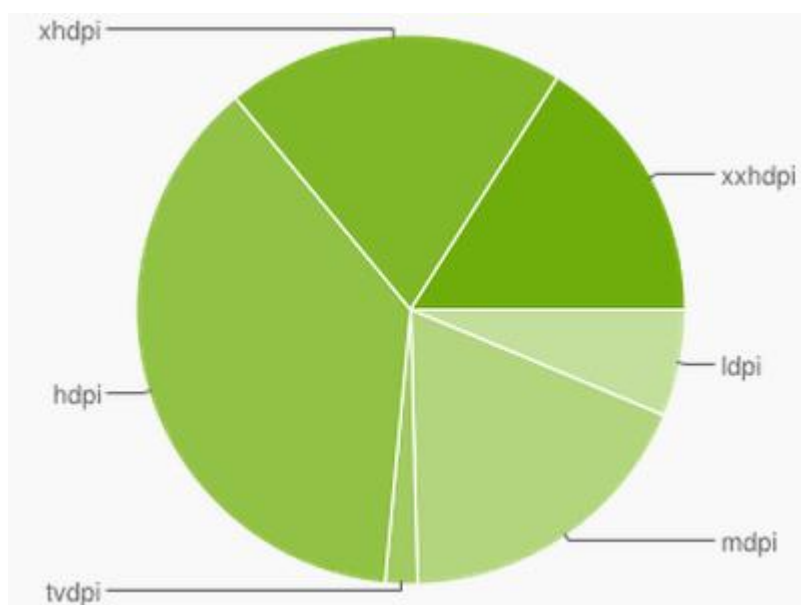


Figura 14: Distribución de dispositivos según densidad de pantalla

La distribución de dispositivos en función de la densidad, a pesar de predominar “hdpi”, es más equitativa sobre todo entre “mdpi”, “xhdpi”, “xxhdpi. Cabe a destacar que la utilización de pantallas de baja y media densidad está decreciendo y la de altas densidades está en auge. No tendremos en cuenta la densidad “tvdpi” ya que está destinada a televisores, y la aplicación no está diseñada para televisores.

La aplicación adaptará las imágenes a cada tipo de densidad de pantalla.

Hay que tener en cuenta que para un mismo tamaño de pantalla podemos encontrar dispositivos con diferentes densidades, la proporción en la actualidad es:

	ldpi	mdpi	hdpi	xhdpi	xxhdpi
Small	5.8%				
Normal		9.9%	36.6%	18.9%	16.0%
Large	0.5%	4.5%	0.6%	0.6%	
Xlarge		3.9%	0.3%	0.5%	

Tabla 7: Distribución de dispositivos según tamaño y densidad

Cómo se puede observar en la tabla, los dispositivos con baja densidad son cada vez menos frecuentes. Prácticamente sólo existen en dispositivos con pantallas pequeñas, estos dispositivos han sido los más difíciles de adaptar ya que son los que ofrecen peores condiciones.

3.3.4 Idioma

La aplicación se ofrece a los alumnos a través de la plataforma Google Play, esta tienda está accesible en muchas partes del mundo. Por ello a pesar de que la aplicación en un principio está destinada a los alumnos de la Universidad Autónoma de Madrid, se ha elegido que el idioma para los contenidos de la aplicación sea el inglés, en concreto el inglés de Estados Unidos.

El inglés ha sido el idioma elegido ya que es la lengua más hablada del mundo, y la que se estudia como segunda lengua en todos los países de habla no inglesa.

Esta elección hace que la aplicación sea útil en todos los países con acceso a Google Play y con conocimiento del inglés. La ficha de publicación en Google Play se ha realizado también en español.

En función del éxito de la aplicación en diferentes países se estudiará la posibilidad de añadir más idiomas.

3.3.5 Resumen de requisitos

- **Sistema Operativo:**
 - Android
- **Versiones del Sistema Operativo:**
 - Desde la versión 2.2 Froyo hasta la 4.4 KitKat
- **Tipos de dispositivos:**
 - Móviles
 - Tablet
- **Tipos de pantalla:**

Los diferenciaremos según:

 - Tamaño: “Small”, “Normal”, “Large” y “Xlarge”.
 - Densidad: “ldpi”, “mdpi”, “hdpi”, “xhdpi” y “xxhdpi”.
- **Idioma:**
 - Inglés

3.4 Limitaciones

Los dieciséis ejercicios que ofrece la aplicación, constarán de un enunciado y una imagen de un circuito MOS, el usuario deberá rellenar una tabla con el estado de los transistores (on, off) y el de la salida, en función de las entradas del circuito.

Las principales limitaciones son:

- Los circuitos tienen que tener un tamaño relativamente pequeño para que se puedan ver en cualquier tipo de pantalla sin necesidad de hacer zoom.
- No se puede ver el circuito y la tabla a la vez.
- Para añadir nuevos ejercicios o preguntas al test hay que actualizar la versión en Google Play.
- No se puede hacer zoom en las imágenes de la aplicación.
- Los circuitos pueden tener un máximo de 4 entradas, para que la tabla quepa en la pantalla.

Exercise 9

D=0

C	B	A	O
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

D=1

C	B	A	O
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Scheme Save

Check Solve Reset

Figura 15: Ejercicio con 4 entradas

- La tabla puede tener como máximo 15 columnas, teniendo en cuenta las entradas, los transistores y la salida, para que se pueda observar correctamente en la pantalla.

Exercise 11

A	B	P1	P2	P3	P4	P5	P6	N1	N2	N3	N4	N5	N6	O
0	0	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	0
0	1	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	0
1	0	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	0
1	1	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	0

Scheme Save

Check Solve Reset

Figura 16: Ejercicio de entradas y 12 transistores

3.5 Módulos de la aplicación

En este apartado se explicarán de manera resumida los diferentes módulos de la aplicación y sus funcionalidades.

3.5.1 Tutorial

La aplicación dispone de un tutorial dónde los alumnos podrán encontrar la teoría relativa a los circuitos MOS, consiste en una sucesión de imágenes que serán útiles para la resolución de los ejercicios.

Las imágenes se muestran a pantalla completa y se podrá pasar hacia delante o hacia atrás deslizando el dedo hacia la izquierda o la derecha, parecido a un libro electrónico.

Las imágenes del tutorial han sido realizadas por el profesor Eduardo Boemo, que se ha basado en los apuntes de clase.

En cualquier momento el usuario puede pulsar el botón atrás de Android volviendo al menú principal de la aplicación.

3.5.2 Ejercicios

En la aplicación hay dieciséis ejercicios de circuitos integrados básicos MOS. Al seleccionar un ejercicio se abrirá una pantalla con su enunciado y la posibilidad de empezar el ejercicio o cargar o borrar soluciones guardadas previamente.

En los ejercicios se pide rellenar una tabla con los estados de los transistores y la salida del circuito en función de las diferentes combinaciones de las entradas.

En la pantalla de resolución del ejercicio se encuentra la tabla a rellenar y la posibilidad de realizar las siguientes acciones:

- Visualizar el esquema: volver a ver el circuito del enunciado.
- Guardar la solución actual en la memoria.
- Corregir el ejercicio: la aplicación indicará si la solución es correcta o no.
- Mostrar la solución del ejercicio.
- Borrar los datos introducidos.

Si el usuario pulsa el botón atrás de Android volverá al menú que muestra todos los ejercicios.

A continuación se resumen los problemas seleccionados y los objetivos docentes de los mismos.

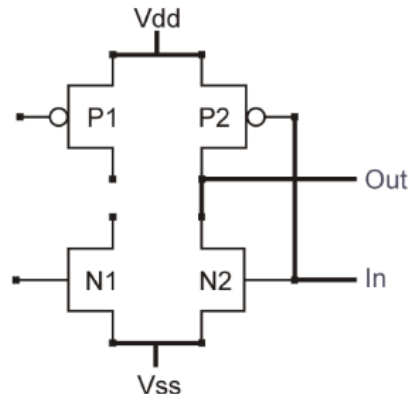
Ejercicio 1
Tipo de problema: Análisis
Concepto: Inversor
Descripción: Configuración básica de un par de transistores P y N tal como se emplazan en un dispositivo tipo Gate Array. Metalizaciones correspondientes a un inversor
Figura: 

Tabla 8: Ejercicio 1

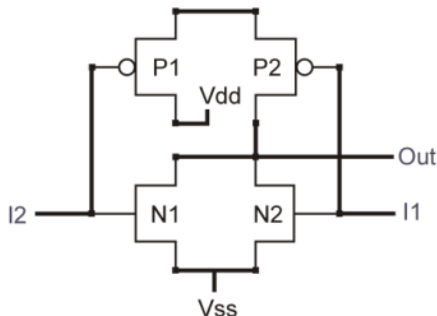
Ejercicio 2
Tipo de problema: Análisis
Concepto: Nor
Descripción: Configuración básica de un par de transistores P y N tal como se emplazan en un dispositivo tipo Gate Array Metalizaciones correspondientes a una puerta NOR
Figura: 

Tabla 9: Ejercicio 2

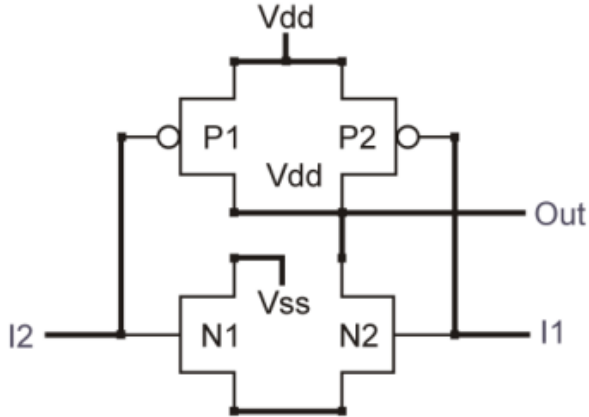
Ejercicio 3
Tipo de problema: Análisis
Concepto: Nand
Descripción: Configuración básica de un par de transistores P y N tal como se emplazan en un dispositivo tipo Gate Array Metalizaciones correspondientes a una puerta NAND
Figura: 

Tabla 10: Ejercicio 3

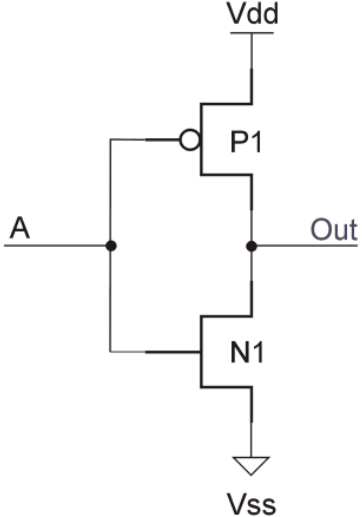
Ejercicio 4
Tipo de problema: Análisis
Concepto: Inversor
Descripción: Inversor. Es una repetición del ejercicio 1 pero dibujado como figura en los libros de texto
Figura: 

Tabla 11: Ejercicio 4

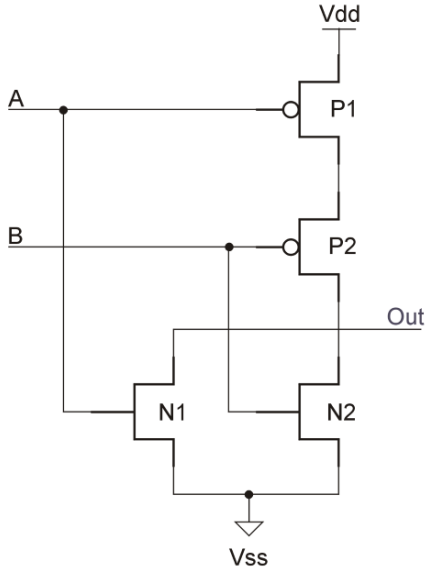
Ejercicio 5
Tipo de problema: Análisis
Concepto: Nor
Descripción: Es una repetición del ejercicio 2, pero dibujado como figura en los libros de texto. Esto facilita la comprensión de la célula.
Figura: 

Tabla 12: Ejercicio 5

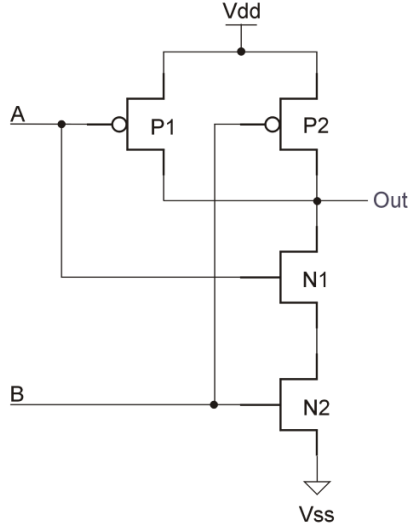
Ejercicio 6
Tipo de problema: Análisis
Concepto: Nand
Descripción: Es una repetición del ejercicio 3, pero dibujado como figura en los libros de texto. Esto facilita la comprensión de la célula.
Figura: 

Tabla 13: Ejercicio 6

Ejercicio 7
Tipo de problema: Análisis
Concepto: Or
Descripción: Concepto de que la OR y la AND tienen más transistores y retardo pues están fabricadas agregando un inversor a la NOR o NAND. Caso correspondiente a la NOR
Figura:

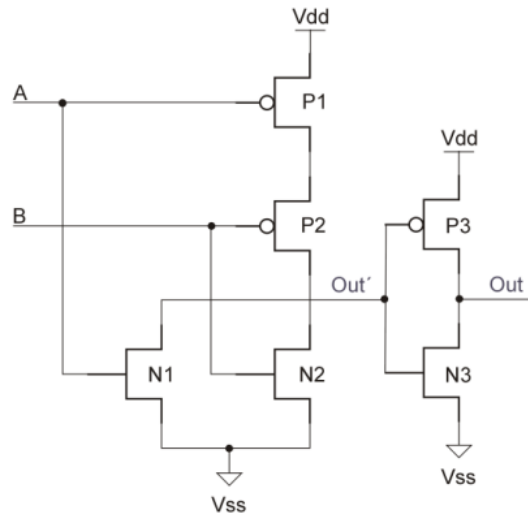


Tabla 14: Ejercicio 7

Ejercicio 8
Tipo de problema: Análisis
Concepto: And
Descripción: Concepto de que la OR y la AND tienen más transistores y retardo pues están fabricadas agregando un inversor a la NOR o NAND. Caso correspondiente a la AND
Figura:

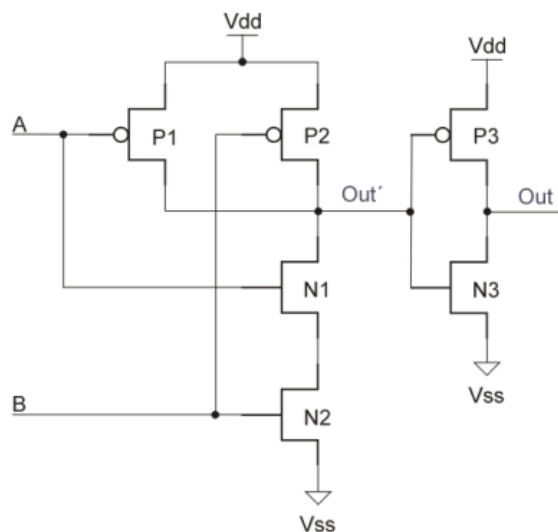


Tabla 15: Ejercicio 8

Ejercicio 9
Tipo de problema: Análisis
Concepto: AOI y OAI
Descripción: La realización de funciones lógicas a partir de una “nube P” de transistores conectada entre VDD y la salida, combinada con una “nube N” de transistores conectadas entre salida y VSS.
Figura:

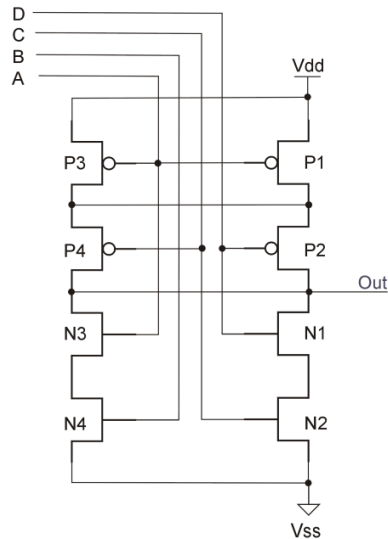


Tabla 16: Ejercicio 9

Ejercicio 10
Tipo de problema: Análisis
Concepto: Xor
Descripción: La XOR a partir de su definición $A \oplus B = A \cdot \overline{B} + \overline{A} \cdot B$ necesita dos AND y una OR. Es decir, $6+6+6=18$ transistores. En este ejercicio se muestra una de las opciones de la XOR con menos transistores. En este caso, 9.
Figura:

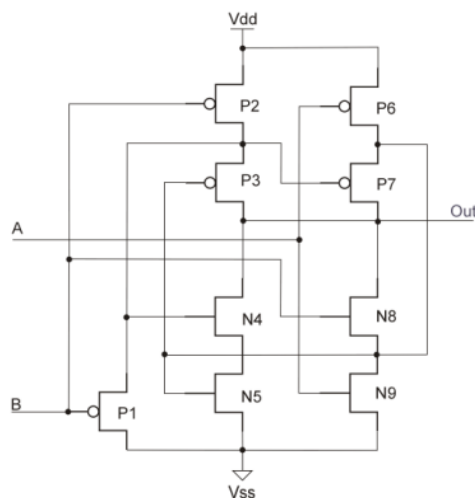


Tabla 17: Ejercicio 10

Ejercicio 11
Tipo de problema: Análisis
Concepto: Xor
Descripción: La XOR a partir de su definición $A \cdot B + \overline{A} \cdot B$ necesita dos AND y una OR. Es decir, $6+6+6=18$ transistores. En este ejercicio se muestra una de las opciones de la XOR con menos transistores. En este caso, 12.
Figura:

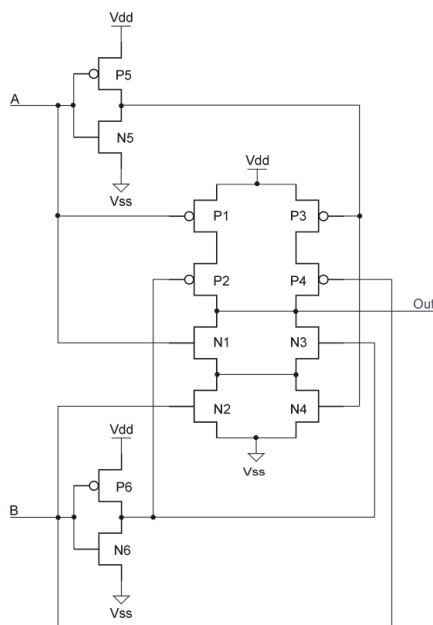


Tabla 18: Ejercicio 11

Ejercicio 12
Tipo de problema: Análisis
Concepto: Mux 2-1
Descripción: Concepto de multiplexor basado en llaves CMOS. De acuerdo con la definición, un multiplexor 2-1 es $A \cdot C + B \cdot C$; es decir, 18 transistores. En el esquema de la figura se realizan la misma función con 6 transistores.
Figura:

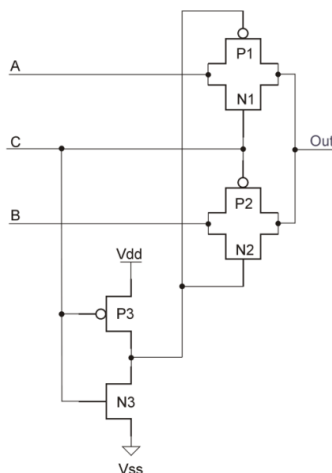


Tabla 19: Ejercicio 12

Ejercicio 13
Tipo de problema: Análisis
Concepto: Función lógica CMOS
Descripción: Es una función lógica a partir de una “nube P” de transistores conectada entre VDD y la salida, combinada con una “nube N” de transistores conectadas entre salida y VSS.
Figura: <p>The diagram shows a CMOS circuit with three inputs: A, B, and C. The PMOS network consists of three transistors: P1 (gate A), P2 (gate B), and P3 (gate C). P1 and P2 are in series, and P3 is in parallel with that series combination. The NMOS network consists of three transistors: N1 (gate A), N2 (gate B), and N3 (gate C). N1 and N2 are in parallel, and N3 is in series with that parallel combination. The output 'Out' is taken from the common drain connection of the PMOS and NMOS networks.</p>

Tabla 20: Ejercicio 13

Ejercicio 14
Tipo de problema: Análisis
Concepto: Función lógica CMOS
Descripción: Es una función lógica a partir de una “nube P” de transistores conectada entre VDD y la salida, combinada con una “nube N” de transistores conectadas entre salida y VSS.
Figura: <p>The diagram shows a CMOS circuit with four inputs: A, B, C, and D. The PMOS network consists of four transistors: P1 (gate B), P2 (gate C), P3 (gate A), and P4 (gate D). P1 and P2 are in series, P3 is in parallel with P1 and P2, and P4 is in series with the entire PMOS network. The NMOS network consists of four transistors: N1 (gate A), N2 (gate C), N3 (gate B), and N4 (gate D). N1 and N2 are in series, N3 is in parallel with N1 and N2, and N4 is in series with the entire NMOS network. The output 'Out' is taken from the common drain connection of the PMOS and NMOS networks.</p>

Tabla 21: Ejercicio 14

Ejercicio 15
Tipo de problema: Análisis
Concepto: Driver
Descripción: Muestra la forma en que se consigue más corriente de salida (<i>driving</i>). Simplemente conectando transistores P y N en paralelo. También se muestra que hace falta un primer inversor para obtener un <i>driver</i> no inversor
Figura:

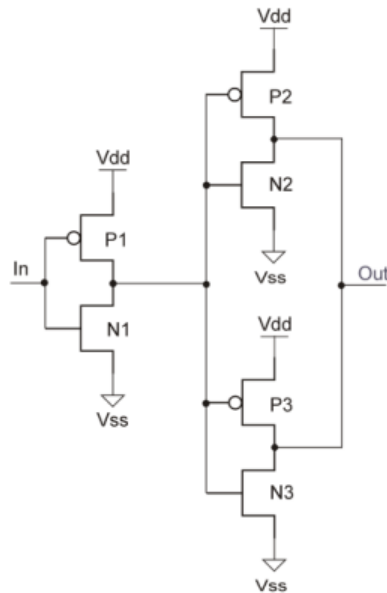


Tabla 22: Ejercicio 15

Ejercicio 16
Tipo de problema: Análisis
Concepto: Driver Buffer Tri-State
Descripción: Célula que combina <i>driver</i> con control de tercer estado.
Figura:

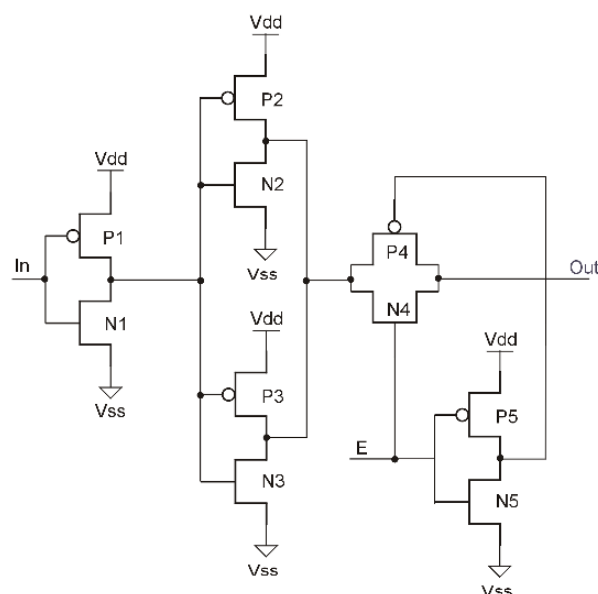


Tabla 23: Ejercicio 16

3.5.3 Guardar

En cualquier momento el usuario podrá guardar la solución introducida. Si es la primera vez que guarda una solución, se creará un directorio en el dispositivo dónde se guardarán todos los ficheros.

Cuando el usuario pulsa el botón “*Save*” se muestra una alerta en la que hay que escribir el nombre con el que se quiere guardar el archivo. Todos los archivos tienen una cabecera automática en la que se indica el número de ejercicio cuya solución se quiere guardar y además, indica si el archivo se está guardando en la memoria externa (EM) o en la memoria interna (IM).

Si el nombre que elegimos para nuestra solución ya existe, saltará una alerta indicándolo y dando la oportunidad de sobrescribir el archivo o cambiar el nombre elegido para guardarlo.

Es necesario guardar una solución para poder enviarla por correo.

3.5.4 Corregir

El usuario podrá comprobar si la solución que ha introducido es correcta pulsando sobre el botón “Check”.

La corrección se puede llevar a cabo en cualquier momento de la resolución, y no sólo indica si la solución es correcta o no, además si la solución introducida no es correcta la aplicación mostrará correcciones más detalladas, indicando en que fila o tabla existe algún error.

3.5.5 Mostrar solución

En la pantalla de resolución del ejercicio, está disponible la opción “*Solve*”. Que mostrará la solución correcta del ejercicio.

3.5.6 Resetear

Opción que permite al usuario borrar los datos que ha introducido, pudiendo así empezar de cero el ejercicio.

Antes de eliminar cualquier solución la aplicación consultará al usuario si está seguro de que desea eliminar los datos introducidos, ya que estos datos se borrarán de manera permanente y no podrán ser recuperados posteriormente.

3.5.7 Cargar y borrar ejercicios

Al pulsar sobre cualquier ejercicio se accede directamente a su enunciado, donde encontramos la posibilidad de iniciar la resolución del ejercicio o de cargar o borrar ficheros con las soluciones guardadas con anterioridad de ese mismo ejercicio.

Al pulsar sobre “*Load or delete*” aparecerá una lista de las soluciones guardadas de ese ejercicio, tanto en la memoria interna como en la externa. Solamente podrá elegir una solución si lo que desea es cargarla, mientras que se pueden eliminar varios ficheros a la vez.

Para borrar soluciones guardadas bastará con seleccionar de la lista de soluciones cuales son las que se quieren eliminar y pulsar el botón “*Delete*”. Saltará un aviso para confirmar que desea borrar los archivos.

Al cargar un archivo la solución del ejercicio guardado se cargará en la pantalla de resolución del ejercicio. De manera que el usuario puede acceder a cualquier solución guardada de forma rápida y sencilla.

3.5.8 Enviar ejercicios

Desde el menú principal se puede acceder a la opción “*Send exercises*”, que permite enviar por correo cualquier solución guardada. Al seleccionar esta opción se mostrará una lista con todos los ejercicios guardados, ya sea en memoria interna o externa.

El usuario elegirá los ejercicios que desee enviar por correo, puede seleccionar todos los que desee, y pulsará el botón para enviar los ejercicios. Se abrirá una alerta para que el usuario seleccione el gestor de correo con el que desea enviar el mensaje. Una vez seleccionado el gestor de correo, el usuario sólo deberá indicar la dirección de correo del destinatario ya que la aplicación automáticamente adjunta los archivos y rellena el asunto y un texto explicativo en el cuerpo del email con las indicaciones necesarias para que el destinatario pueda abrir los ficheros recibidos con nuestra aplicación.

Instrucciones que recibe el destinatario en el cuerpo del email:

- Descargar los archivos adjuntos, normalmente se descargarán en la carpeta /sdcard/download del dispositivo.
- Mover estos archivos a la carpeta /sdcard/MOSCircuits, para ello utilizar un gestor de archivos. (Existen aplicaciones Android que actúan como gestor de archivos.)

* El destinatario debe haber instalado y ejecutado la aplicación MOSCircuits, para que la carpeta /sdcard/MOSCircuits se haya creado.

Si el usuario realiza estos pasos correctamente, al abrir la aplicación tendrá los archivos cargados.

Esta funcionalidad facilita al usuario compartir las soluciones con otros alumnos o con el profesor de una forma rápida y sencilla.

3.5.9 Test

Se ha añadido un test, para que los alumnos puedan comprobar sus conocimientos teóricos. Consiste en 12 preguntas con 4 posibles respuestas, la aplicación ofrece la posibilidad de corregir el test, informando la puntuación obtenida y cuáles son las preguntas que ha fallado.

3.5.10 Manual de ayuda

La aplicación dispone de un pequeño manual de usuario, en el que se explica cómo completar una tabla, guardar ejercicios, cargar ejercicios o enviarlos por correo. La aplicación es muy sencilla e intuitiva por lo que este manual es suficiente para poder sacar el máximo partido a la aplicación.

Se puede acceder a este manual desde el menú principal de la aplicación o desde cualquier pantalla de la aplicación pulsando sobre el botón “Menú” de Android y seleccionando la opción “*Help*”.

3.5.11 Botón Menú de Android

Este botón está activo desde todas las pantallas de la aplicación, exceptuando las pantallas Help y About, que son precisamente las que encontramos si pulsamos sobre el botón Menú de Android.

- Help: se accede al manual de usuario descrito en el apartado 3.5.10.
- About: acceso a la pantalla “Acerca de” de la aplicación, dónde se encuentran los datos relativos a los desarrolladores del proyecto.

4 Desarrollo

En este apartado vamos a explicar de manera detallada la parte más técnica del proyecto, describiendo cómo están programados los principales módulos de la aplicación.

4.1 Primeros pasos

Dado la falta de experiencia con las herramientas de desarrollo de Android y el lenguaje de programación Java, se han realizado las siguientes tareas:

- Lectura y estudio de varios libros de programación Java.
- Lectura de un tutorial de XML.
- Realización de varios tutoriales de Android.
- Realización de cursos de Android online y un curso ofrecido por la Universidad Autónoma de Madrid.
- Estudio de la API de Android, para conocer sus características y funcionalidades principales.
- Realización de pequeñas aplicaciones para familiarizarse con el entorno de desarrollo de Android.

Una vez familiarizados con el entorno y los lenguajes de programación, comenzamos a desarrollar el proyecto.

4.2 Herramientas utilizadas

Para el desarrollo de la aplicación se han utilizado las siguientes herramientas:

- Ordenador personal
- Eclipse
 - o SDK de Android
 - o Simulador ADT Android
- Dispositivos Android (móviles y tabletas)
- Herramientas de edición de imágenes

4.3 Tipos de archivo

A continuación se van a exponer los tipos de archivo que puede contener una aplicación Android.

4.3.1 Activity

Una actividad es una clase en java que representa cada una de las pantallas de una aplicación, y su principal objetivo es interactuar con el usuario. Desde el momento que una actividad aparece en la pantalla hasta que desaparece pasa por varias etapas, lo que se conoce como ciclo de vida de una actividad.

Las actividades están formadas por una parte lógica y una parte gráfica. La parte lógica es un archivo java, que contiene la clase que permite manipular, interactuar y desarrollar el código de esa actividad. Mientras que la gráfica es un archivo XML, también conocido como layout, en el que declaramos todos los elementos que se mostrarán por pantalla.

Todas las actividades se inician con el método “OnCreate”, desde dónde se enlazan la parte lógica y la gráfica mediante “SetContentView()”. Para poder utilizar en la parte lógica los elementos de la parte gráfica es necesario recuperar su identidad, “findViewById()”.

Existe una pila de actividades que se utiliza para organizarlas: cuando una actividad se inicializa pasa a ocupar la parte superior de la pila y se convierte en la actividad en marcha, la actividad anterior se mantiene por debajo en la pila, y cuando el usuario utiliza el botón “Back” del dispositivo vuelve a aparecer en pantalla.



Figura 17: Botón “Back”

Una actividad tiene esencialmente cuatro estados:

- Una actividad que está en primer plano en la pantalla, está **activa** o ejecutándose.
- Una actividad que todavía es visible pero que no está en primer plano, está en **pausa**. Podría ser finalizada por el sistema si se necesitase memoria.

- Una actividad que ha sido eclipsada por otra actividad, está en **stop**. Todavía mantiene todos los estados y la información, pero no es visible por el usuario por lo que el sistema la finalizará si necesita memoria.
- **Destruída**: Una actividad que está en pausa o en stop puede ser eliminada por el sistema. Cuando se vuelva a mostrar al usuario el sistema tendrá que reiniciarla o restaurarla al estado anterior.

El siguiente diagrama muestra el ciclo de vida de una actividad.

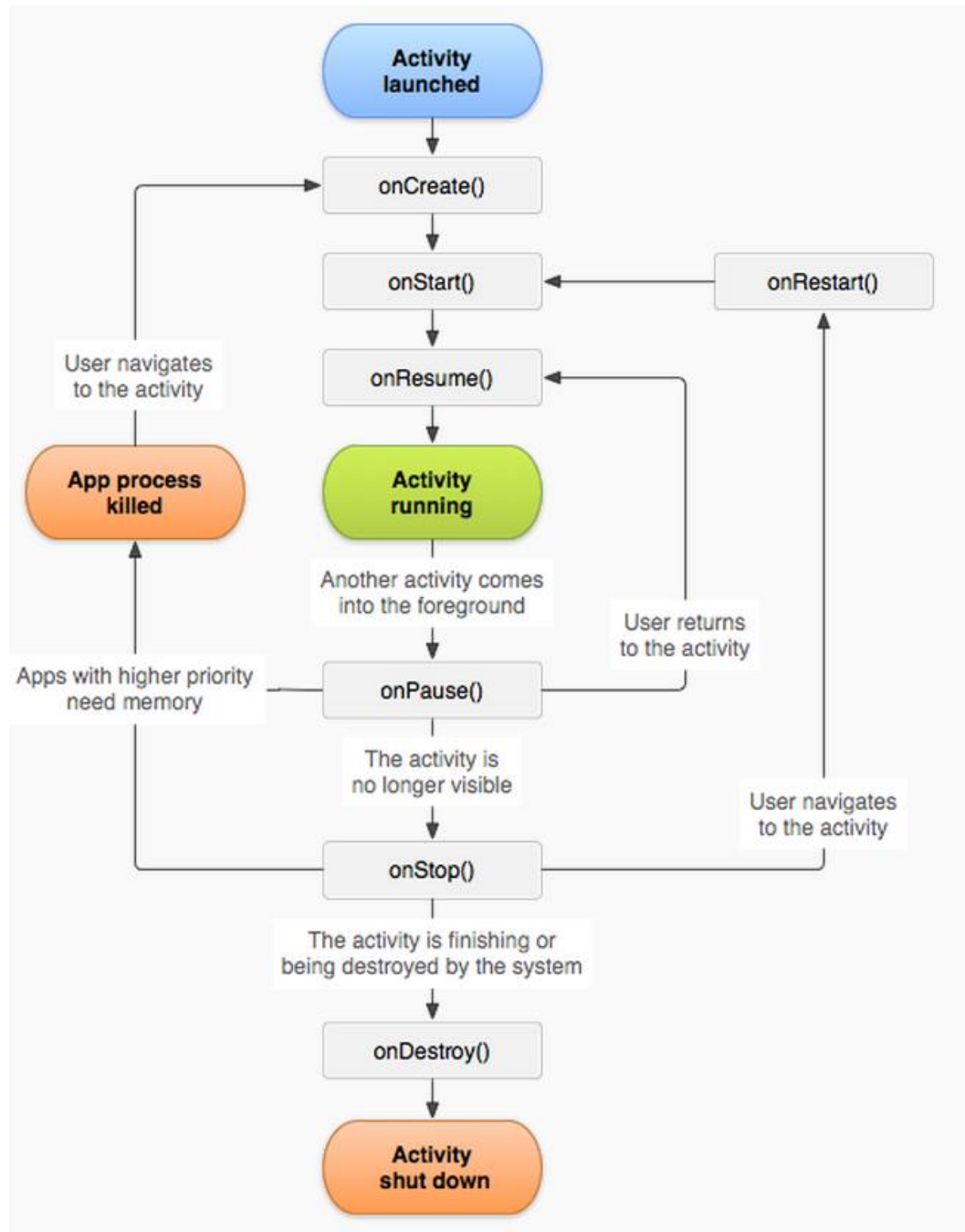


Figura 18: Ciclo de vida de una actividad.

4.3.2 Views

Esta clase representa la vista que contiene los componentes de la interfaz de usuario. Ocupa un área rectangular en la pantalla y es responsable de dibujar y manejar los eventos de una aplicación. Es la clase básica de los widgets, que se usan para crear componentes interactivos (botones, campos de texto, etc.).

4.3.3 Manifest

Todas las aplicaciones tienen que tener un archivo `AndroidManifest.xml` en la raíz de su directorio, este archivo contiene información esencial acerca de la aplicación. Información que el sistema tiene que tener antes de poder ejecutar el código de la aplicación.

Algunas de las funciones del `AndroidManifest.xml` son:

- Dar nombre al paquete java de la aplicación, que sirve como identificador.
- Definir todas las actividades de la aplicación y algunas de sus características.
- Declarar los permisos que la aplicación debe tener la aplicación para ejecutarse. En nuestro caso sólo necesita permisos para escribir en la memoria externa.
- Declara la versión mínima y máxima de Android en que se podrá ejecutar la aplicación.
- Indica el número de versión del código de la aplicación.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.MOSCircuits"
    android:installLocation="auto"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="7"
        android:targetSdkVersion="19" />

    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.MOSCircuits.MainActivity"
            android:label="@string/app_name"
            android:screenOrientation="portrait" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name="com.MOSCircuits.About"
            android:label="About"
            android:screenOrientation="portrait" >
        </activity>
```

Figura 19: `AndroidManifest.xml`

4.4 Esquema general

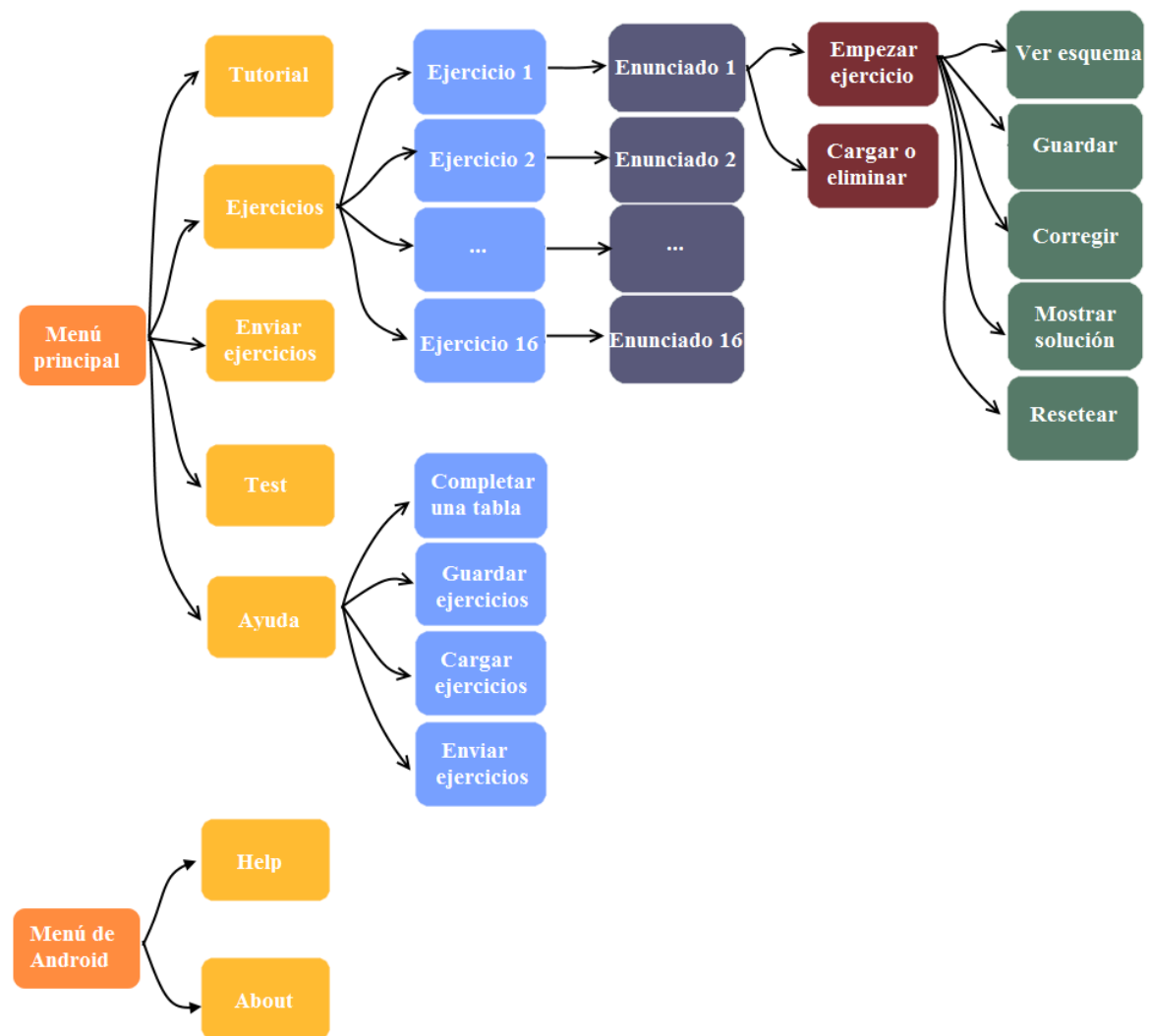


Figura 20: Esquema general de la aplicación

En este esquema se muestran los principales módulos de la aplicación, más adelante iremos explicando cada uno de ellos. Aunque sólo se han puesto los módulos del primer ejercicio, el resto de ejercicios también disponen de ellos.

Hay que destacar que en nuestro proyecto las actividades no son gestionadas por la pila, sino que cada vez que lanzamos una nueva actividad finalizamos la actividad actual. Además gestionamos el botón “Back” de Android, estableciendo a qué actividad debe volver el sistema, si se pulsa el botón, según el módulo en el que se encuentre. Excepto con los botones del menú de Android, que permitimos que la pila guarde la actividad desde la que se accede al botón, de manera que cuando pulsemos “Back” volveremos a dónde estábamos.

4.5 Menús de la aplicación

La aplicación consta de cuatro menús:

- Menú principal: desde el menú principal se puede acceder al tutorial, ir al menú de ejercicios, enviar ejercicios, ir al test o al menú de ayuda.

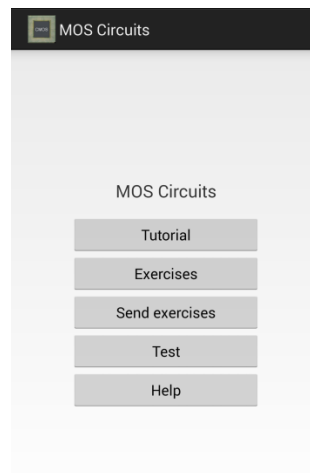


Figura 21: Menú principal

- Menú de ejercicios: a este menú se accede desde el menú principal, desde aquí se puede ir a cualquiera de los dieciséis ejercicios disponibles en la aplicación. En esta pantalla ha sido necesario poner una vista de tipo “ScrollView” para poder ver todos los botones del menú, ya que este tipo de vista permita realizar desplazamientos en la pantalla.

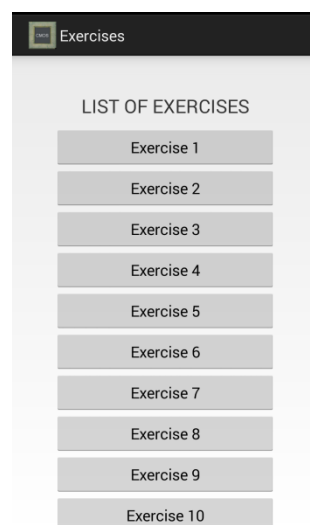


Figura 22: Menú ejercicios

- Menú de ayuda: es el manual de usuario y se puede llegar hasta él desde el menú principal o el botón “Menú” de Android, dónde se explica cómo se ha de completar una tabla y cómo guardar, cargar o enviar ejercicios.

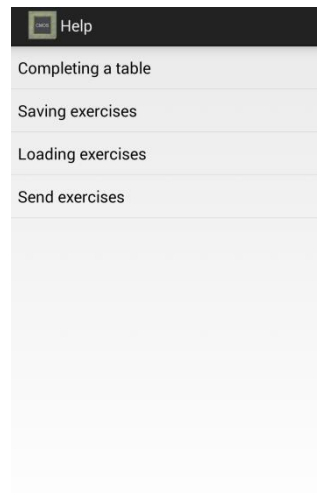


Figura 23: Menú ayuda

- Menú al que se accede desde el botón “Menú” de Android: este botón está disponible en todas las pantallas, y ofrece dos opciones, el menú de ayuda y la pantalla “acerca de” (“About”). Dentro de estas dos opciones no está habilitado el botón menú. Este menú aparece en la parte inferior de la pantalla desde la cual se accede.

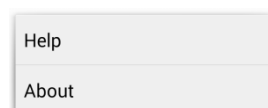


Figura 24: Menú accesible desde el botón “menú” de Android

Estos menús se han diseñado de manera muy simple, se trata de un conjunto de botones que al pulsarlos ejecutan una acción programada. En la mayoría de los casos se lanza una actividad diferente por cada botón, menos en el caso del menú ayuda, que todos los botones lanzan la misma actividad, ya que en las opciones de este menú sólo se muestran vistas. Otra excepción es el caso de “Enviar ejercicios” del menú principal, que lanza un “AlertDialog”.

4.6 Tutorial

El tutorial está formado por un conjunto de fotos en formato “.png”, que se van mostrando en la pantalla simulando un pequeño libro electrónico.

Se desarrolla en una única vista de tipo Java, dónde se registran los identificadores de todas las imágenes, que se muestran una a una por pantalla.

Las imágenes se seleccionan según un contador que varía en función del avance o retroceso de las páginas del tutorial. Es un proceso cíclico, ya que cuando se llega a la última imagen del tutorial si seguimos avanzando se mostrará la primera página. Para pasar de una página a otra basta con deslizar el dedo hacia la izquierda, para avanzar.

Nada más entrar en el tutorial, se lanza un aviso que explica cómo pasar las páginas.

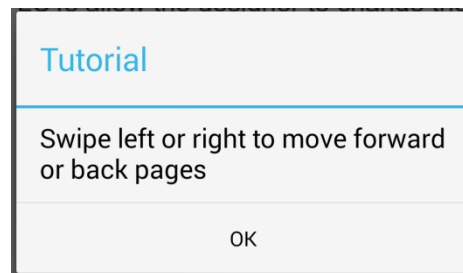


Figura 25: Aviso tutorial

4.7 About

Esta actividad fue de las primeras en desarrollarse dado su simplicidad, sigue estrictamente el estilo de las aplicaciones anteriores del DSLab. Es la actividad “acerca de”, en ella se incluye información básica del proyecto.

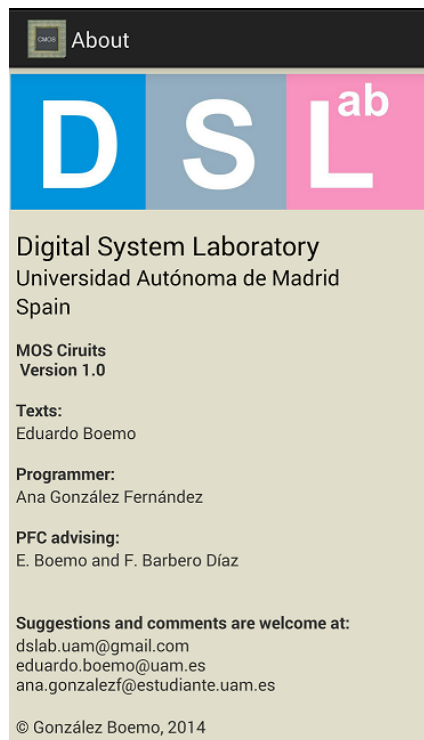


Figura 26: Pantalla “Acerca de”

4.8 Diseño y desarrollo de ejercicios

Cada uno de los ejercicios está formado por la vista de la actividad (archivo “.java”) y la propia actividad (archivo “.xml”). Para explicar el diseño y desarrollo de los ejercicios nos vamos a centrar en cada una de ellas por separado.

4.8.1 Vistas

El lenguaje utilizado para desarrollar la vista de la actividad es XML.

Lo primero que se hace al desarrollar una vista es definir el tipo de Layout que vamos a utilizar, que establece la manera en que se van a representar los Widgets en nuestra vista. En nuestro caso se ha utilizado “LinearLayout (vertical)”, lo que hace que los elementos aparezcan uno debajo de otro.

Para formar la tabla, añadimos dentro del LinearLayout un elemento tabla (TableLayout) y dentro de este, varios de tipo fila (TableRow) hasta completar el tamaño de la tabla deseado. En cada fila definiremos elementos de texto (TextView) para establecer las posibles entradas del circuito y botones (ToggleButton) necesarios para poder resolver el ejercicio.

Los botones escogidos para las celdas de la tabla, han sido del tipo `ToggleButton` ya que permite cambiar entre dos estados. Para el estado de los transistores hemos establecido que los `ToggleButton` estén por defecto en “Off”, si pulsamos se pone en “On”, si se vuelve a pulsar cambia a “Off” y así sucesivamente. Mientras que para la salida del circuito el valor por defecto es “0” y al pulsar cambia a “1”.

También se utilizan botones del tipo “`Button`” para activar las funcionalidades del ejercicio.

Es en esta vista dónde se define el tamaño y tipo de letra, se establecen los márgenes, la alineación de los elementos... y demás aspectos visuales. Además se asigna un identificador a cada botón para que pueda ser nombrado desde la Actividad, para los `ToggleButton` los identificadores utilizados han sido TB01, TB02, etc. Mientras que los “`Button`” tienen asignados un identificador que especifica la funcionalidad del botón, por ejemplo: `boton_guardar`.

Cuando se pulsa alguno de los botones lanzamos una función, dependiendo del botón que sea se lanzará una función u otra, para ello utilizamos el atributo “`android: onClick`”. Lo igualamos al nombre del método, que estará definido en la actividad, y al pulsar el botón se llamará a dicho método.

Como máximo nuestra función puede tener cuatro entradas, para que se visualice correctamente en la pantalla. De hecho si el circuito tiene cuatro entradas en nuestra vista tendremos que definir dos tablas en vez de una. A continuación se muestran dos vistas de ejercicios, la primera corresponde a un circuito con cuatro entradas y la segunda a uno con tres.



Figura 27: Vistas ejercicios 3 y 4 entradas

En todas las tablas la primera fila es de texto, ya que se corresponde con el encabezado de la tabla. Las siguientes representan los datos del ejercicio, los elementos con fondo gris son texto, que representan las combinaciones posibles de las entradas, y los elementos de la zona azul son los “ToggleButton” con los que se resolverá el ejercicio.

En las dos últimas filas de la vista están los botones de funcionalidades del ejercicio, del tipo “Button”.

4.8.2 Actividad

Es en la Actividad donde se definen todas las variables, y se desarrollan los métodos que proporcionan las funcionalidades que ofrece el ejercicio.

Como se ha explicado antes cada “ToggleButton” de la vista tiene un identificador, que usaremos para manejar los datos que introducen los usuarios. Para ello declaramos dos vectores:

- Uno del tipo “ToggleButton”, cuyo tamaño es igual al número de “ToggleButton” definidos en la vista. Y se usará para asignar un “ToggleButton” a cada elemento

del vector creado. Si modificamos algún elemento de este vector también se modificará en la vista.

- El segundo vector es del tipo “Boolean”, se ha elegido este tipo ya que es un tipo de dato similar al “ToggleButton” y es fácil trabajar con él. Su tamaño es igual que el del vector anterior y en la actividad trabajaremos con él, ya que la vista no cambia al modificarlo.

A continuación se crea una función para que cada vez que el usuario pulse un “ToggleButton”, el nuevo estado se guarde. Llamamos a esta función desde la vista con el atributo explicado con anterioridad, “android: onClick”. Estos datos también se guardan en el archivo de preferencias, que se explicará con más detalle a continuación.

Una vez tenemos los datos de la tabla disponibles en la actividad, podemos ejecutar las diferentes funciones que ofrece la aplicación:

Mostrar esquema

En esta función se llama a otra actividad con su correspondiente vista, dónde se muestra la imagen del circuito del enunciado.

Guardar

Este botón llama a otra actividad, que guarda en un archivo la solución introducida por el usuario.

En esta actividad está definido un vector del tipo “Boolean” para cada uno de los ejercicios que se usa para guardar los datos introducidos por el usuario, menos para el ejercicio 16 que es un vector del tipo “int”, este ejercicio se explicará más adelante.

La vista de esta actividad se presenta en forma de diálogo, dónde se muestra una cabecera que contiene el número de ejercicio y la memoria donde se va a guardar, IM si se guarda en la memoria interna (parte de la memoria interna reservada para datos de la aplicación) y EM si se guarda en la memoria externa, que no tiene por qué ser una tarjeta SD, los móviles más actuales habilitan una parte de memoria interna para que esté disponible para el usuario y también se llama memoria externa. En este dialogo también se pide al usuario que introduzca el nombre con el que quiere guardar el archivo.

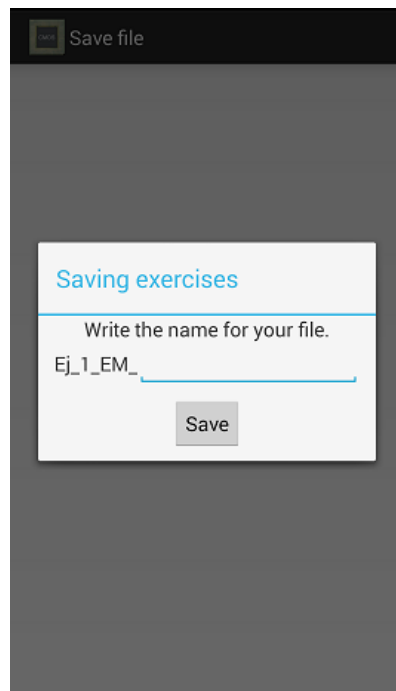


Figura 28: Diálogo Guardar ejercicio

Si ya existe un archivo con el nombre que ha introducido el usuario, se lanzará un aviso informando de que ya existe un archivo con ese nombre, y se preguntara si desea sobrescribir el ejercicio.

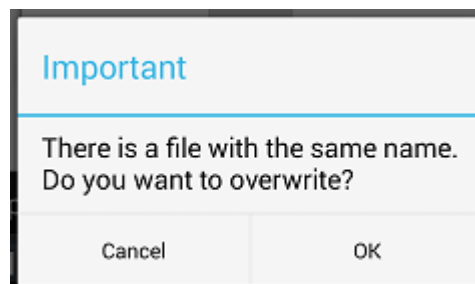


Figura 29: Sobrescribir ejercicio

Corregir

Para comprobar si la solución introducida es correcta, definimos dos vectores por cada fila de la tabla del ejercicio. Un vector con los datos introducidos por el usuario y otro con la solución correcta correspondiente a dicha fila. Como las tablas tienen solución única bastará con comparar estos vectores, y se mostrará por pantalla mediante un mensaje “Toast” el número de filas incorrectas, en caso de que haya algún error o el mensaje que confirma que la solución es correcta.

En los ejercicios con cuatro entradas, en vez de crear dos vectores por fila se crean dos vectores por tabla, y al corregir el ejercicio se indicará la tabla en la que hay errores, en vez de la fila.

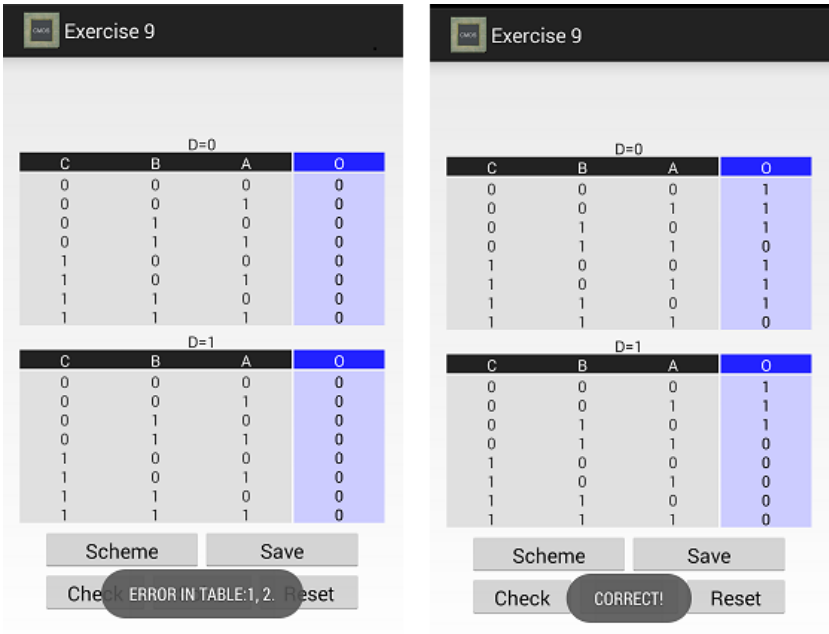


Figura 30: Corregir ejercicio

Mostrar solución

Función que muestra la solución en la vista. Para ello hemos definido un vector al principio de la Actividad con la solución final del ejercicio, y con el método “setChecked()” igualamos los valores de los ToggleButton de la vista a los de la solución correcta. Además guardamos los datos en las preferencias. Y mediante un mensaje “Toast” informamos al usuario de que se está mostrando la solución del ejercicio.

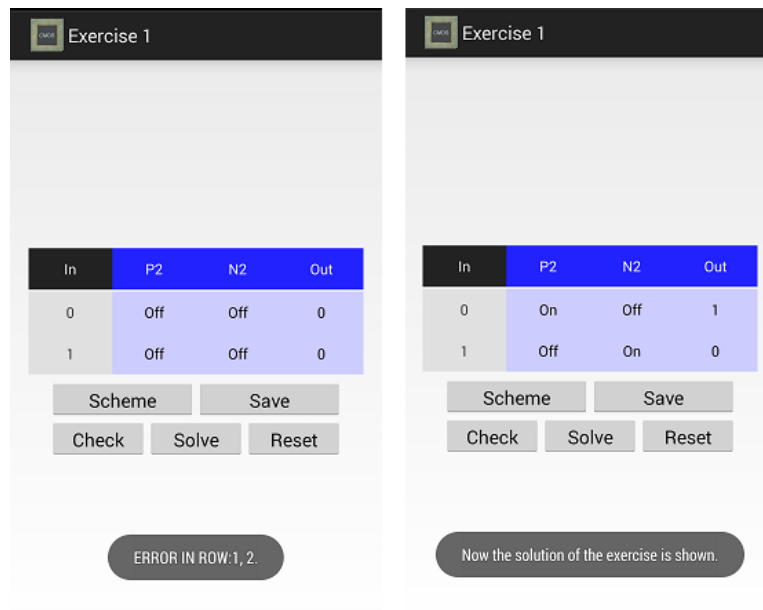


Figura 31: Mostrar solución del ejercicio

Resetear

Al pulsar sobre el botón “reset” se lanza un aviso para confirmar que el usuario quiere borrar los datos introducidos. Si el usuario confirma que desea resetear el ejercicio, usamos el método “setChecked()” para poner todos los “ToggleButton” de la vista a su valor por defecto (Off o 0 según corresponda). Si el usuario cancela el aviso, el sistema vuelve al ejercicio. Además se guardan los datos en el vector booleano y en las preferencias.

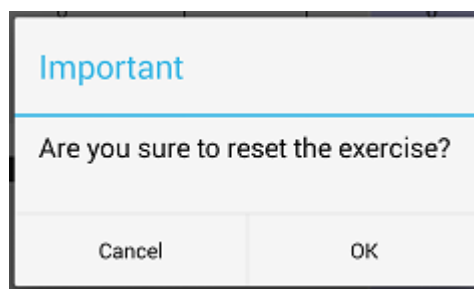


Figura 32: Aviso confirmación borrar ejercicio

4.9 Ejercicio 16

Este ejercicio es diferente al resto ya que la solución al circuito presentado puede ser 0, 1 o Z. Por lo que no podemos utilizar el tipo de botón que hemos utilizado para el resto de ejercicios. Cómo no había ningún botón predeterminado que permitiese cambiar entre tres estados, hemos creado una clase nueva a la que hemos denominado como “TriToggleButton”. Usamos “ToggleButton” para indicar el estado de los transistores y “TriToggleButton” para indicar la solución al circuito.

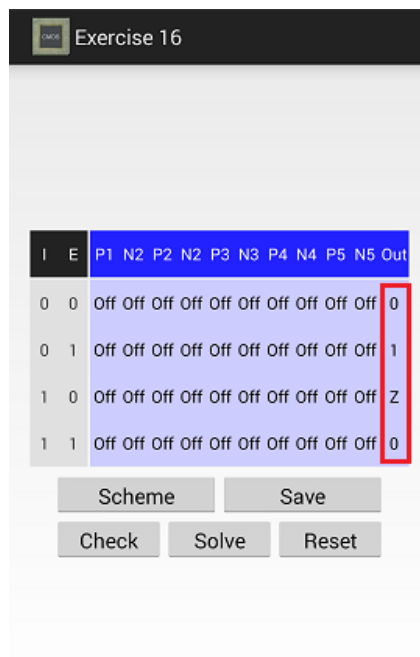


Figura 33: Ejercicio 16

Es por esta razón por la que no podemos utilizar vectores de tipo Boolean, y usamos vectores de tipo “int”, para guardar las preferencias y para corregir y mostrar la solución del ejercicio. En este ejercicio la corrección de la solución se realiza igual que en los anteriores, comparando vectores y mostrando mediante un mensaje “Toast” el número de fila en el que hay errores.

4.10 Preferencias

Las preferencias no son más que datos que se guardan para ser compartidos con otros métodos o actividades. Consisten en uno o más archivos “XML” en los que las actividades pueden guardar información, los datos se guardan en un formato determinado: etiqueta_valor.

Cada actividad que use preferencias debe declarar un objeto de tipo “SharedPreferences”, e indicar con que nombre se guarda el archivo de preferencias y en qué modo de acceso. Nosotros hemos elegido el modo: `MODE_PRIVATE`, que solo permite a nuestra aplicación acceder a las preferencias. Para poder guardar datos en un archivo de preferencias hay que declarar un editor de preferencias e indicar el tipo de dato que se guarda y su valor. Mientras que para leer un dato de un archivo de preferencias usaremos uno de los métodos proporcionados por la clase “SharedPreferences”, `getBoolean` o `getInt`, según corresponda.

Estos ficheros son accesibles desde cualquier actividad de la aplicación, y podrán ser consultados aunque se salga de la aplicación o se apague el dispositivo ya que el fichero no se borra.

En nuestra aplicación hemos usado las preferencias principalmente en tres ocasiones:

- En la actividad de cada enunciado, para guardar el número de ejercicio en el que estamos. Que será utilizado posteriormente en la actividad que carga y borra los ficheros con las soluciones guardadas, correspondientes a cada ejercicio.
- En la actividad de cada ejercicio, guardando los datos introducidos por el usuario. Lo primero que se hace al lanzar un ejercicio es cargar las preferencias, de manera que se cargarán los datos introducidos en la última sesión del ejercicio.
- En el test, se guardan las respuestas introducidas por el usuario.

De manera que todos los datos introducidos por el usuario estarán guardados en las preferencias, y podrán ser consultados en cualquier momento.

4.11 Cargar o eliminar soluciones guardadas

Esta funcionalidad está desarrollada en una actividad propia (sin vista asociada), que puede ser lanzada desde cualquier enunciado de un ejercicio. Al igual que en la actividad para guardar ejercicios se declaran tantos vectores booleanos como ejercicios, cada uno con el tamaño correspondiente a su tabla para poder cargar los datos de cada ejercicio. A excepción del ejercicio 16 que el vector es de tipo `int`.

Mediante el método de las preferencias se obtiene el número de ejercicio desde el que hemos lanzado la actividad, y se muestra en una alerta una lista con todos los ficheros

guardados para dicho ejercicio. El usuario podrá cargar sólo una de las soluciones guardadas, y se lanzará la actividad del ejercicio con los datos del fichero cargados. Para ello primero se lee el fichero, se guardan los datos en el vector y después se carga el vector en las preferencias del ejercicio correspondiente. Como la actividad de cada ejercicio carga las preferencias al iniciarse, los datos se cargan automáticamente, sobrescribiendo los datos introducidos en la última sesión del mismo.

Sin embargo esta actividad permite seleccionar todos los archivos si lo que se desea es eliminarlos. Antes de borrarlos la actividad lanza una alerta de confirmación, ya que los archivos se borran de manera permanente.

4.12 Enviar soluciones

Sólo se puede acceder a esta opción desde el menú principal y su funcionalidad está desarrollada en la actividad del menú principal de la aplicación. Permite enviar una o más soluciones guardadas por correo electrónico.

Al pulsar sobre la opción “Send exercises” del menú principal se lanza una alerta donde se muestra una lista con todas las soluciones guardadas. El usuario debe seleccionar las que quiere enviar y al pulsar sobre “continuar”, la aplicación le dará a elegir con qué gestor de correo electrónico quiere enviar el email.

Una vez elegido el gestor con el que se desea enviar el email, el usuario sólo deberá indicar el email del destinatario, ya que la actividad rellena los siguientes campos del email:

- Asunto del correo: MOS exercises
- Cuerpo del correo: donde se explica al destinatario del email cómo utilizar los ficheros, en los siguientes pasos:
 - o Descargar los ficheros, generalmente se descargan en /sdcard/download.
 - o Copiar estos ficheros en la carpeta de la aplicación (/sdcard/MOSCircuits) usando un gestor de contenidos.
- Además adjunta los ficheros seleccionados.

Para poder copiar los ficheros en la carpeta de la aplicación, el destinatario debe tener instalada la aplicación y haber guardado por lo menos una vez un ejercicio. Cuando haya realizado estos pasos, los ficheros descargados estarán disponibles en la opción “Load or delete” de los ejercicios correspondientes.

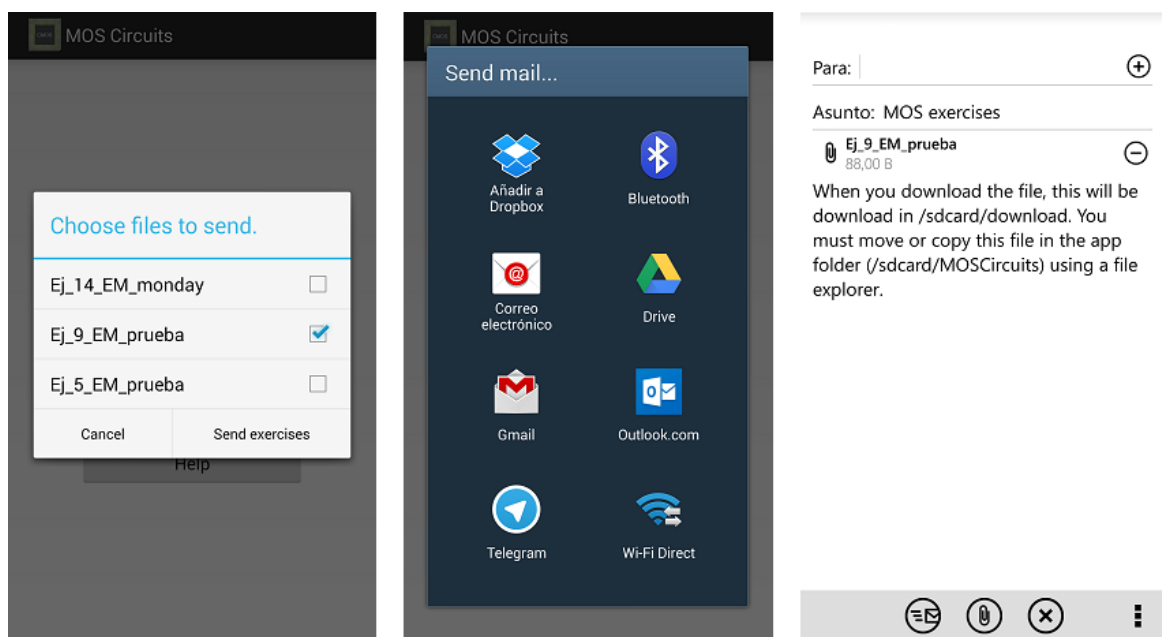


Figura 34: Enviar solución por correo

4.13 Test

El test se ha desarrollado en una actividad con su correspondiente vista asociada. En la vista usamos un tipo de vista “ScrollView” para poder mover la pantalla arriba y abajo y ver todas las preguntas. Es dónde definimos cada pregunta del test, usamos un TextView para el enunciado de la pregunta y un RadioGroup con cuatro RadioButton, para las posibles respuestas. Hemos decidido usar este tipo de botón ya que las preguntas sólo tienen una solución correcta, y lo que se buscaba es que el usuario solo pudiera seleccionar una respuesta. Además añadimos un botón (“Button”), que permita corregir el test.

En la actividad se define un vector “RadioButton”, el cuál modificará la vista si cambiamos su valor. Y un vector booleano que es el que utilizaremos para manejar los datos. En esta actividad se desarrollan dos funcionalidades principalmente:

- Guardar los datos introducidos por el usuario en las preferencias
- Corregir el test: para corregir el test creamos un vector booleano por cada pregunta, y le asignamos las respuestas correctas. Por lo tanto solo tendremos que comparar pregunta a pregunta los vectores, de manera que se informa al usuario qué pregunta es la que ha fallado. Además de utilizar un contador, para indicar al usuario la puntuación obtenida en el test.

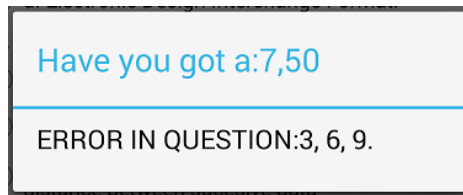


Figura 35: Puntuación test y preguntas erróneas

4.14 Adaptar a los diferentes tipos de pantalla

Queremos que la aplicación esté disponible en el mayor número de dispositivos posibles, para ello es necesario adaptar las vistas a los diferentes tamaños y densidades de pantalla.

Es difícil hacer una vista que se vea correctamente en pantallas grandes y pequeñas, es por eso que Android proporciona una manera sencilla de diseño, permite crear carpetas de layouts para cada tamaño de pantalla. En nuestro caso hemos creado una carpeta para cada tamaño de pantalla que maneja Android:

- Layout-small
- Layout
- Layout-large
- Layout-xlarge

Por lo que hemos desarrollado cada vista cuatro veces, en ellas se realizan cambios como el tamaño de letra, posición y tamaño de los botones, o el tamaño de los márgenes.

Android proporciona una herramienta para simular un dispositivo de cualquier tamaño y densidad, lo que permite comprobar cómo se visualizarán las diferentes vistas en diferentes “terminales”. En nuestro caso hemos ejecutado la aplicación en dispositivos con diferentes tamaños y densidades de pantallas, para asegurarnos de que se visualizan bien.

En el caso del tutorial, como está compuesto de imágenes la adaptación es diferente, se realiza directamente desde la actividad. Se crea una clase de tipo “View” y mediante el método “OnDraw” se elabora la vista, para ello se define un lienzo de la clase “canvas”.

Sobre este lienzo se dibujan, con la función “DrawBitmap”, las imágenes del tutorial, estableciendo el bit del eje “x” y del eje “y” en el que empieza la imagen para centrarla.

Para poder adaptar las imágenes a las diferentes pantallas, lo primero que hay que hacer es leer la relación de la pantalla en la que se ejecuta la aplicación, cuanto más cercano sea este valor a 1 más cuadrada será la pantalla. Si la relación es mayor de 1.52 la imagen se mostrará sin modificar, ya que se visualizará adecuadamente. Mientras que si la relación de la pantalla es menor a 1.52, habrá que reducir el tamaño de la imagen para que se vea correctamente en la pantalla, se ha comprobado que es suficiente con reducir un 25%.

Para estos cálculos se han tenido en cuenta los márgenes, la barra de estado de Android y la del título de la actividad, ya que reducen el espacio disponible para dibujar la imagen.

4.15 Diagrama y descripción de clases

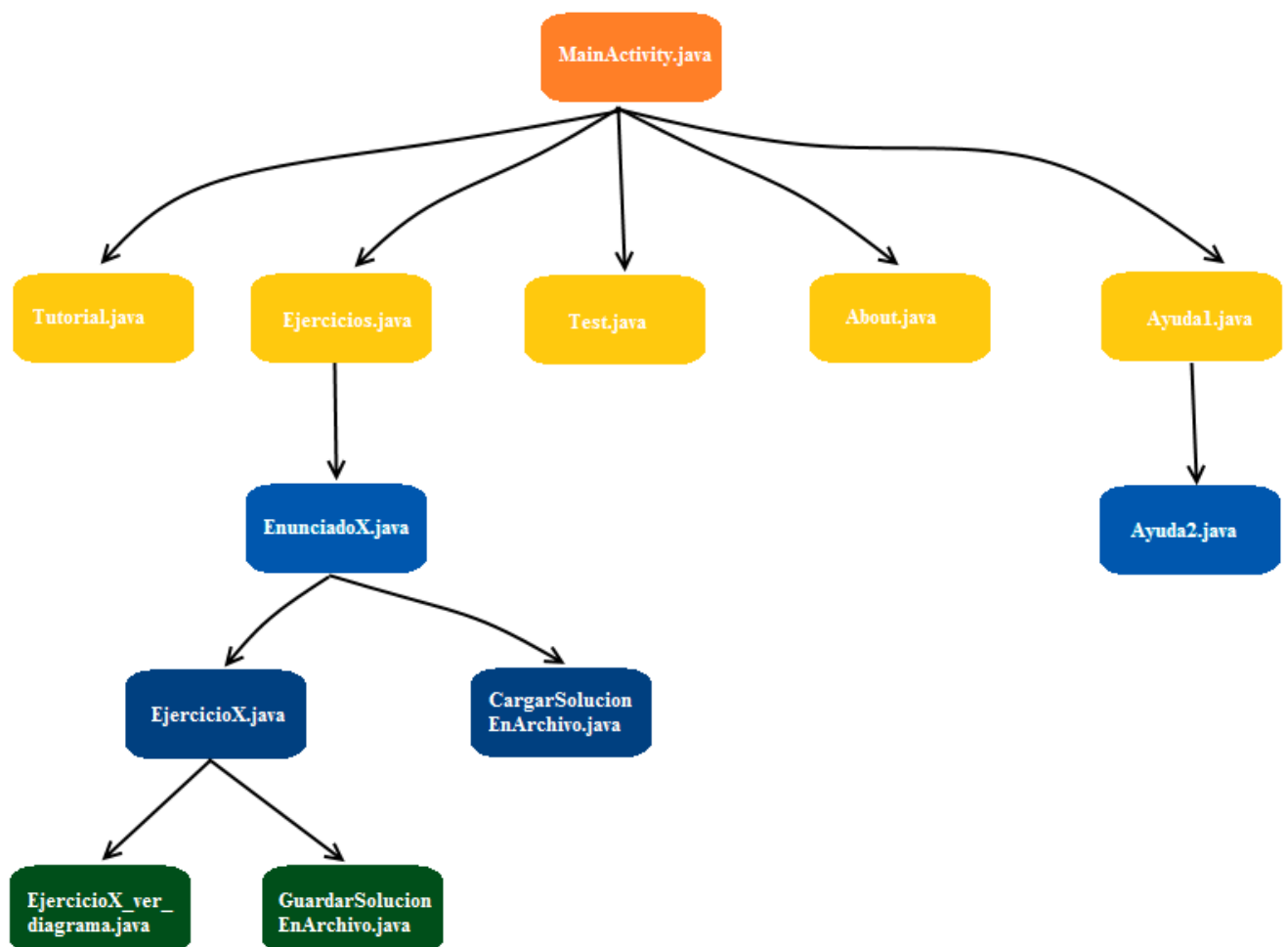


Figura 36: Diagrama de clases

A continuación vamos a describir brevemente las clases principales.

- **MainActivity.java:** ofrece el menú principal y desde ahí se realiza el envío de soluciones por correo electrónico.
- **About.java:** Muestra los datos relativos al “acerca de” de la aplicación.
- **Tutorial.java:** muestra el conjunto de imágenes que forma el tutorial, se realiza la adaptación de estas imágenes a las diferentes pantallas y se gestiona el avance y retroceso de las páginas del tutorial.
- **Test.java:** Se muestran las preguntas del test y se corrigen indicando la puntuación obtenida.
- **Ayuda1.java:** ofrece el menú con las ayudas disponibles.
- **Ayuda2.java:** Visualiza la ayuda seleccionada en el menú anterior (Ayuda1).
- **Ejercicios.java:** ofrece el menú de ejercicios, y lanza el enunciado del ejercicio seleccionado.
- **EnunciadoX.java:** muestra el enunciado del ejercicio y permite acceder al ejercicioX.java o a GuardarSolucionEnArchivo.java.
- **CargarSolucionEnArchivo.java:** desde aquí se cargan las soluciones guardadas en la actividad del ejercicio correspondiente, también permite borrar las soluciones guardadas.
- **EjercicioX.java:** desde aquí se resuelve el ejercicio y se accede a las distintas funcionalidades disponibles.
- **EjercicioX_ver_diagrama.java:** muestra la imagen del circuito del enunciado, para facilitar al usuario su resolución.
- **GuardarSolucionEnArchivo.java:** guarda el resultado introducido por el usuario en un fichero, impidiendo que existan dos ficheros con el mismo nombre.
- **TriToggleButton.java:** Clase en la que se desarrolla el botón de tres estados necesario para el ejercicio 16.

4.16 Descripción métodos principales

Método	Descripción	Actividad
onCreate()	Método que se invoca al iniciar la actividad, en él se desarrollan todas las acciones que tienen que ejecutarse nada más iniciar la actividad.	Todas
mostrarTutorial()	Lanza la actividad tutorial.java	MainActivity.java
lanzarHelp()	Lanza el menú de ayuda, Ayuda1.java	MainActivity.java
lanzarEjercicios()	Lanza el menú de ejercicios, Ejercicios.java	MainActivity.java
lanzarTest()	Lanza la actividad Test.java	MainActivity.java
lanzarEnviarEjercicios()	Muestra todas las soluciones guardadas en una alerta y permite enviarlas por correo.	MainActivity.java
onCreateOptionsMenu()	Crea un menú de opciones para el botón “menú” de Android.	Todas
onOptionsItemSelected()	Se establecen las opciones que se van a mostrar en el menú de Android.	Todas
onKeyDown()	Indica que hará el sistema cuando se pulse el botón “back” de Android, en este caso regresará a la actividad anterior.	Todas
onTouchEvent()	Detecta los gestos realizados al tocar la pantalla.	Tutorial.java
onDraw()	Método para diseñar la vista, dibuja las imágenes del tutorial adaptadas a la pantalla.	Tutorial.java
lanzarEjX()	Lanza la actividad correspondiente al enunciado en el que estamos, EjercicioX.java	EnunciadoX.java
lanzarCargarEjX()	Lanza la actividad para cargar o borrar los archivos guardados del ejercicio, CargaSolucionEnArchivo.java	EnunciadoX.java
Guardar_RB()	Guarda los datos cada vez que se pulsa un RadioButton.	Test.java
Guardar_TB()	Guarda los datos cada vez que se pulsa un ToggleButton.	EjercicioX.java
Guardar_Preferencias()	Guarda los datos del ejercicio en el fichero correspondiente de preferencias.	EjercicioX.java Test.java
Cargar_Preferencias()	Carga en el ejercicio los datos guardados en las preferencias.	EjercicioX.java Test.java
verDiagrama()	Muestra la imagen del circuito del enunciadoX.	EjercicioX.java
Save()	Lanza la actividad encargada de guardar la solución, GuardarSolucionEnArchivo.java	EjercicioX.java
Comprobacion()	Comprueba si los datos introducidos por el usuario son correctos y muestra un “Toast” indicando el resultado de la comprobación.	EjercicioX.java
ayuda()	Muestra la solución del ejercicio X en la tabla.	EjercicioX.java
resetear()	Borra los datos introducidos por el usuario en el ejercicio actual.	EjercicioX.java
Guardar()	Método encargado de guardar los datos de cada ejercicio en el archivo correspondiente.	GuardarSolucionEnArchivo.java

Tabla 24: Descripción métodos principales

5 Integración, pruebas y resultados

5.1 Pasos previos a la publicación en Google Play Store

Una vez la aplicación está terminada y probada concienzudamente, se procede a la publicación en Google Play Store, pero antes hay que exportar la aplicación firmada, y es conveniente alinear el archivo “.apk”.

5.1.1 Firmar una Aplicación

Las principales razones por las que debemos firmar nuestra aplicación son:

- Como medida de seguridad y requisito de garantía.
- Para poder distribuir e instalar la aplicación sin problemas.
- Para que sólo nosotros podamos modificar y actualizar nuestra app.
- Porque es un requisito que exige el Android Market para subir aplicaciones.

Los pasos a seguir para firmar la aplicación:

1. Abrimos eclipse y seleccionamos el archivo *AndroidManifest.xml*, del proyecto que queremos firmar, ubicándonos en la pestaña *Manifest*, en la sección *Exporting*.

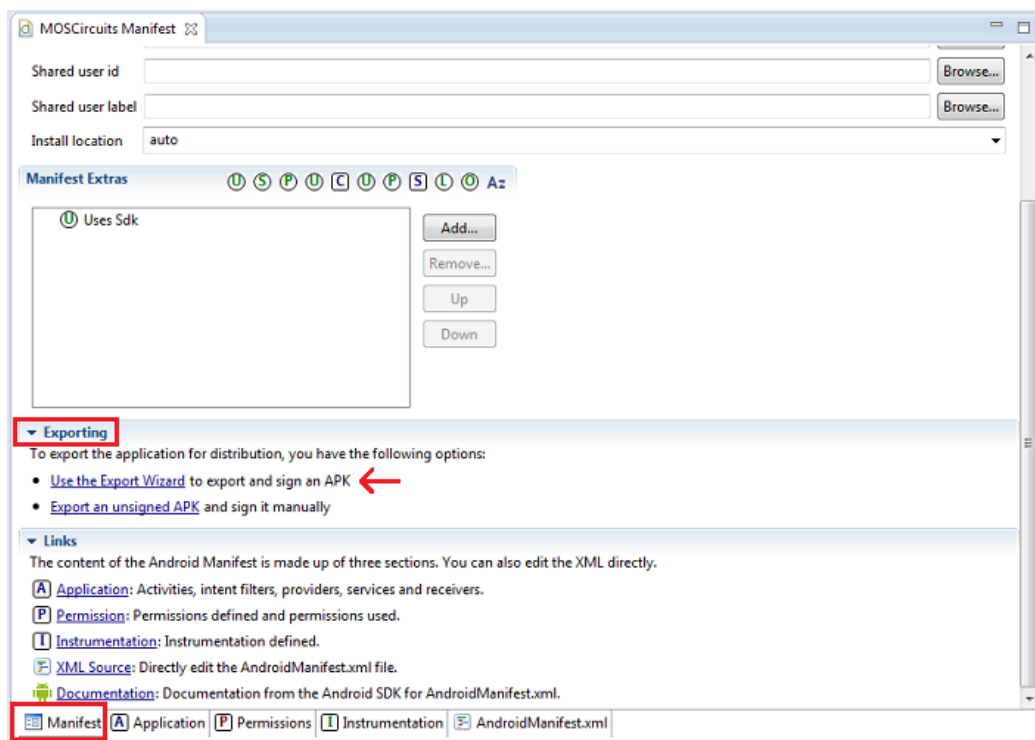


Figura 37. Firmar aplicación: Use the Export Wizard

2. Seleccionamos la opción *Use the Export Wizard*, para poder realizar el proceso de una manera sencilla. En la primera pantalla tenemos que ver seleccionado de manera automática nuestro proyecto. Si no hay ningún error clicamos en el botón *Siguiente*.

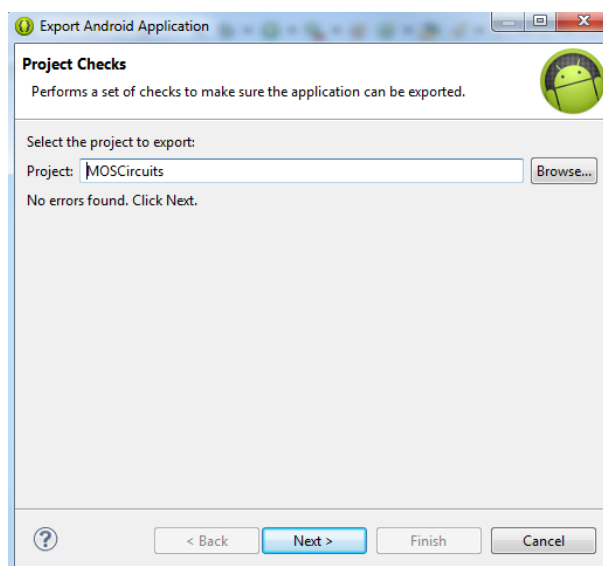


Figura 38. Firmar Aplicación: seleccionar proyecto a exportar

3. Para firmar una aplicación es necesario tener una *keystore*, que es un almacén de claves en dónde se encuentran todos los certificados validados que se nos pedirán. La primera vez que firmamos nuestra aplicación deberemos crear una *keystore*. Por lo tanto seleccionamos la opción *Create new keystore* y llenamos los campos según la siguiente información:

- Location: directorio de nuestro ordenador dónde queremos que se guarde la *keystore* y le damos el nombre que queramos.
- Password: nuestra *keystore* deberá tener una contraseña con un mínimo de 6 dígitos.

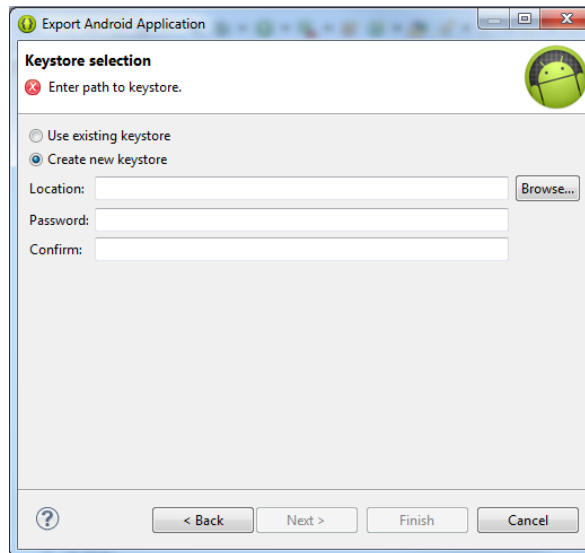


Figura 39. Firmar Aplicación: keystore

Una vez confirmada la contraseña damos clic en el botón *Siguiente*.

4. Formulario para asignar datos a nuestra keystore.

- Alias: puede ser el mismo que el del nombre o una abreviación del mismo.
- Contraseña: nuevamente asignamos una contraseña y la confirmamos.
- Validity (years): definimos la duración de la validación de nuestra keystore en años.
- Los siguientes campos corresponden a información personal y de la organización

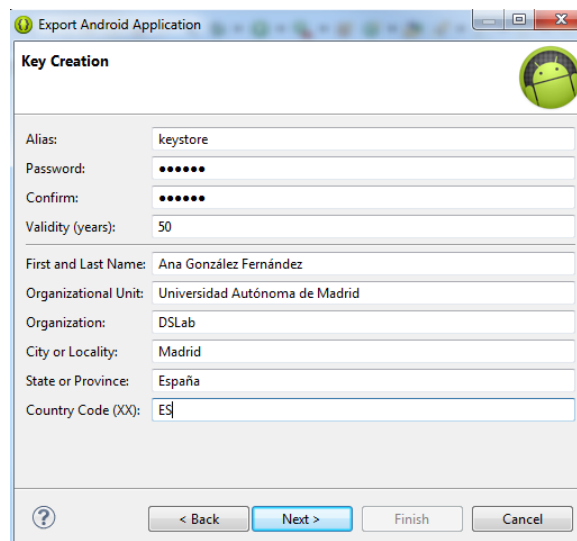


Figura 40. Firmar Aplicación: Key creation

5. Elegimos el directorio de destino dónde queremos que nos guarde el archivo .apk firmado y también nos indicará cuándo expira el certificado. Clicamos en *Finalizar* y ya tengo la aplicación firmada.

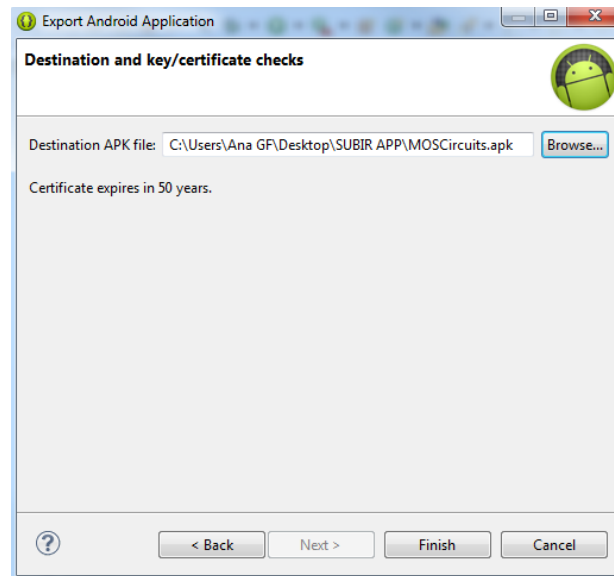


Figura 41. Firmar Aplicación: destino final proyecto.apk

Para firmar futuras aplicaciones podemos usar la *keystore* que he creado para firmar esta aplicación.

5.1.2 Alineamiento del “.apk”

El alineamiento se realiza como algo opcional, para mejorar el rendimiento de la aplicación, optimizando los paquetes “.apk” adaptándolos a los requisitos óptimos del sistema Android. Lo que garantiza que todos los datos sin comprimir empiezan con una particular alineación de 4 bytes con respecto al comienzo del archivo, lo que proporciona una optimización de rendimiento cuando se instala la aplicación en un dispositivo Android.

Esta alineación permite al sistema Android acceder directamente a la posición en la que se encuentran los datos, minimizando el uso de RAM.

La lectura de recursos en una aplicación no alineada será más lenta y requerirá mucha memoria. En el peor de los casos la instalación de varias aplicaciones no alineadas podría aumentar los requisitos de memoria y sobrecargar el sistema, por lo que el dispositivo será muy lento y tendrá un consumo de batería excesivo.

Para alinear el “.apk” abrimos el cmd, cambiamos el directorio a la carpeta en la que se encuentra nuestro proyecto ya firmado y ejecutamos la siguiente línea de código:

```
C:\Users\Ana GF\Desktop\APP SUBIR>zipalign -v 4 ProyNoAlign.apk ProyAlign.apk
```

Figura 42: Alineamiento: zipalign

5.2 Publicación en Google Play Store

Una vez tenemos la aplicación firma y alineada, la subimos a la tienda de aplicaciones de Android.

5.2.1 Cuenta de desarrollador

Si es la primera vez que subimos una aplicación será necesario darse de alta como desarrollador de Android, para ello habrá que pagar 25\$, rellenar un formulario de datos y elegir un nombre de desarrollador.

En este caso he usado la cuenta del laboratorio de la universidad, que ya se ha utilizado para publicar otros proyectos y que se usará en futuros proyectos ya que no caduca. El nombre de desarrollador es “DSLab UAM”.

5.2.2 Subir una aplicación

Para añadir una aplicación entramos con nuestro usuario en Google Play Store Developer Console, si la aplicación es nueva clicamos sobre *+Añadir nueva Aplicación*. Damos un nombre a nuestra aplicación y procedemos a rellenar la ficha de la aplicación:

- Título: MOS Circuits.
- Descripción Breve: Tutorial teórico y ejercicios sobre circuitos integrado básicos MOS.
- Descripción completa: MOS Circuits, es una aplicación educativa que ofrece un determinado número de problemas sobre circuitos integrados MOS.

El objetivo principal es que el usuario pueda estudiar y resolver problemas desde un dispositivo Android. También se indica si el ejercicio se ha resuelto correctamente y la solución al problema.

Dispone de un tipo test para evaluar los conocimientos.

El usuario puede guardar las soluciones de los ejercicios y enviarlas por email, de manera que el receptor puede cargar las soluciones en su aplicación MOS Circuits. Contiene un manual básico en el que explica cómo utilizar la aplicación.

App gratuita.

- Capturas de pantalla de diferentes tamaños.
- Icono de alta resolución en formato 512x512 píxeles.
- Imagen destacada de la aplicación.
- Tipo de aplicación: aplicación.
- Categoría: Educación.
- Clasificación del contenido: Nivel de madurez alto.
- Sitio web: http://arantxa.ii.uam.es/~euroform_dslab/android.htm
- Correo electrónico: dslab.uam@gmail.com
- Política de privacidad

Además hay que subir el archivo “.apk”, que es un archivo comprimido que contiene la aplicación (firmada y alineada), y a partir del cual se realiza la instalación de la misma.

Por último establecemos el precio y la distribución, en este caso la aplicación es gratuita. Y ha sido distribuida a todos los países disponibles en la plataforma (138 + España).

La aplicación ya está disponible a través de la aplicación Google Play Store o en este enlace:

<https://play.google.com/store/apps/details?id=com.MOSCircuits>

5.2.3 Actualizar una aplicación

A pesar de haber realizado pruebas exhaustivas en la aplicación, siempre hay que corregir pequeños errores o realizar mejoras, por lo que es necesario actualizar la aplicación.

Para ello es necesario aumentar el número de versión de código en el AndroidManifest.xml, ya que si no la plataforma de subida no aceptará el fichero “.apk”. El procedimiento es sencillo, repetiremos el proceso de exportación y subiremos el fichero “.apk” actualizado, firmado y alineado, desde Google Play Developer Console.

Hasta el momento solo hay dos versiones, en la segunda versión se ha introducido el tutorial, márgenes en el test y se han corregido algunos errores en los ejercicios 5, 7 y 8.

5.3 Estadísticas proporcionadas por Google Play Store

Una vez publicada la aplicación en Google Play Store podemos acceder a la gran cantidad de información que ofrece la consola de desarrollador, acerca de la aplicación. Esta información se actualiza diariamente. Las estadísticas ofrecidas a continuación han sido recogidas el 9 de diciembre de 2014, un mes después de la publicación de la aplicación.

5.3.1 Instalaciones totales de la aplicación

A continuación se muestra una gráfica en la que se puede observar el número de instalaciones totales por usuario:

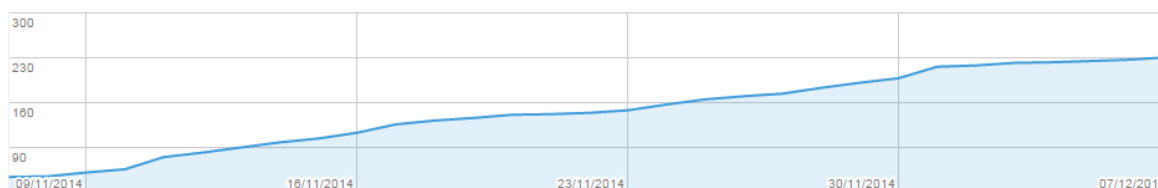


Figura 43: Descargas totales de la aplicación

En total la aplicación se ha descargado 231 veces, de las cuales 149 siguen activas.

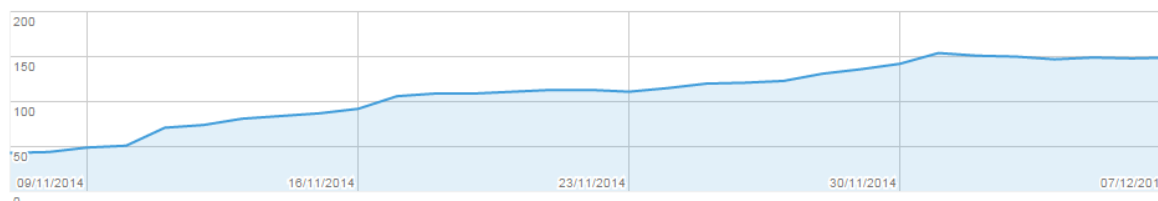


Figura 44: Instalaciones actuales a 09/12/2014

La aplicación tiene una media de 45 instalaciones por semana y 17 desinstalaciones.

5.3.2 Instalaciones según versión de Android

En la siguiente tabla se muestra el número de descargas de la aplicación según la versión de Android:

Versión	Número de descargas
Android 4.4	90
Android 4.1	37
Android 4.2	37
Android 4.3	23
Android 2.3.3-2.3.7	21
Android 4.0.3-4.0.4	12
Android 5.0.0	7
Android 2.1-2.2	4

Tabla 25: Número de descargas según versión de Android

Instalaciones totales según versión de Android

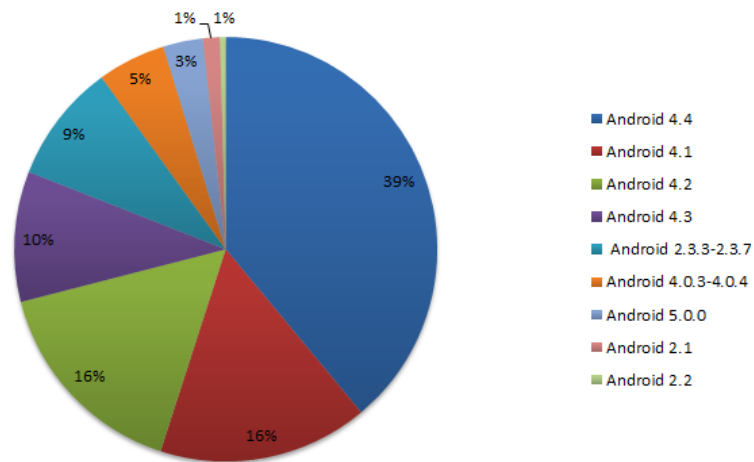


Figura 45: Descargas totales según versión de Android

Como era de esperar hay un mayor número de descargas en las versiones más recientes.

5.3.3 Instalaciones totales por países

Esta gráfica muestra los países en los que se ha descargado la aplicación y el número de veces que se ha descargado:

Instalaciones por país

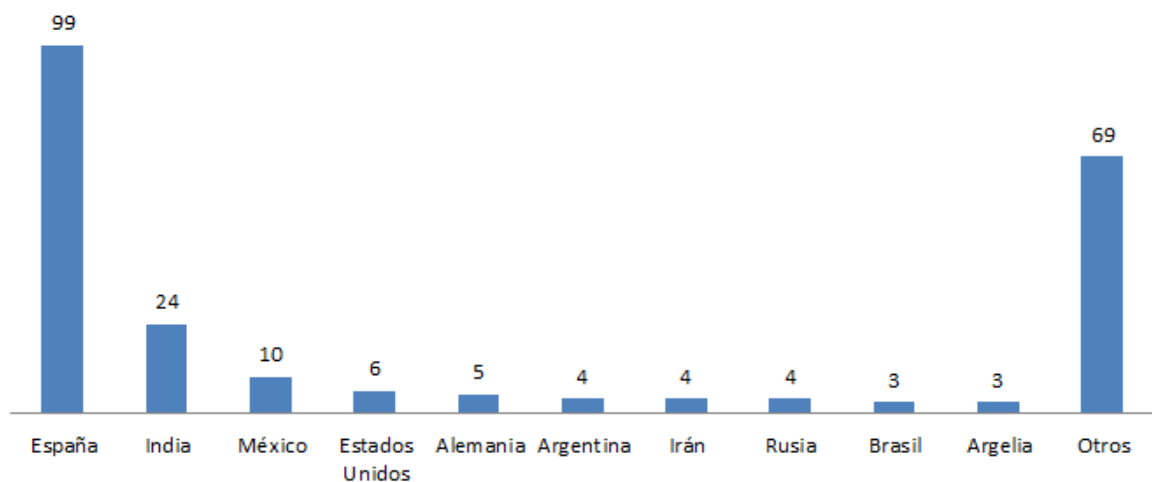


Figura 46: Instalaciones totales por país

6 Conclusiones y trabajo futuro

6.1 Conclusiones

El proyecto ha cumplido los objetivos principales marcados al comienzo del mismo, y se han desarrollado algunas funcionalidades adicionales que no estaban previstas.

El objetivo principal del proyecto era aprovechar la alta penetración de los móviles y tabletas entre los estudiantes, para ofrecer a los alumnos una aplicación educativa en la plataforma Android. Esta aplicación permite a sus usuarios estudiar y realizar la guía de problemas de la asignatura Dispositivos Integrados Especializados, desde sus dispositivos móviles.

Las principales funcionalidades y tareas que se han desarrollado en la aplicación son:

- Un tutorial teórico
- Manual de usuario para facilitar el uso de la aplicación
- Dieciséis ejercicios de circuitos MOS adaptados a un dispositivo móvil
- Herramientas de comprobación de errores de los ejercicios y para solucionarlos
- Posibilidad de resetear un ejercicio
- Guardar las soluciones introducidas por los usuarios
- Cargar o eliminar los archivos con las soluciones guardadas
- Enviar por correo las soluciones
- Test de respuesta múltiple que permita al alumnado medir sus conocimientos
- Herramientas de corrección del test, con un contador que informa de la nota obtenida
- Adaptar imágenes y layouts a los diferentes tipos de pantalla

Desde el punto de vista didáctico el objetivo era aprender a programar aplicaciones para dispositivos Android, sobre todo:

- Lenguaje Java
- Lenguaje XML
- Entorno de desarrollo Eclipse
- Arquitectura de Android
- Publicar aplicaciones en Google Play Store

- Organización y planificación de un proyecto

El tiempo empleado en desarrollar el proyecto han sido 540 horas, repartidas de la siguiente manera:

- Aprendizaje de los lenguajes de programación y el entorno de trabajo, Eclipse: 80h.
- Desarrollo de la aplicación y realización de pruebas: 350h.
- Memoria: 110h.

La aplicación desarrollada es estable, y en los dos meses que lleva en Google Play no se ha reportado ningún error.

6.2 Encuesta

Esta aplicación ha sido utilizada por los alumnos que han cursado este año la asignatura Dispositivos Integrados Especializados, y se les ha realizado una encuesta para que valoren algunos puntos de la aplicación y poder comprobar si la aplicación es útil.

	Muy malo (1)	Malo (2)	Normal (3)	Bueno (4)	Muy bueno (5)
Organización general					
Aspecto de los gráficos					
Facilidad de uso					
Utilidad de la opción enviar					
Utilidad de la Ayuda General (F1)					
Fiabilidad y estabilidad					
Consumo de batería					
Velocidad					
Ordenación de los ejercicios					
Comentarios y sugerencias:					

Tabla 26: Encuesta

La encuesta se realizó el día 2 de diciembre de 2014, después del examen. Fueron 27 los alumnos que respondieron a la encuesta, la mayoría de ellos no han utilizado la opción de enviar por correo, ni el menú de ayuda disponible, por lo que no hemos podido analizar el impacto de estas dos funcionalidades.

Tras analizar las encuestas los resultados obtenidos han sido los siguientes:

Organización general

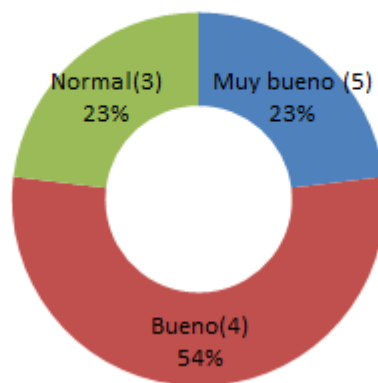


Figura 47: Encuesta Organización general

Aspecto de los gráficos

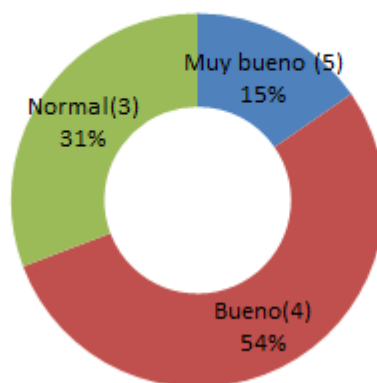


Figura 48: Encuesta Aspecto de los gráficos

Facilidad de uso

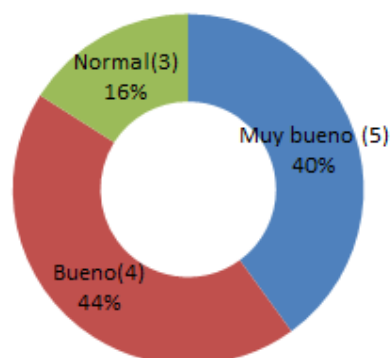


Figura 49: Encuesta Facilidad de uso

Fiabilidad y estabilidad

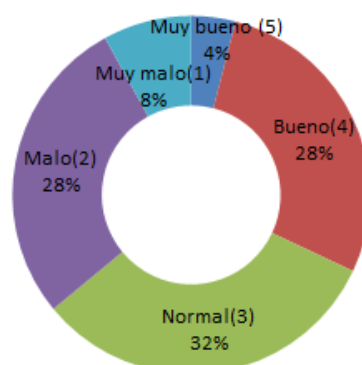


Figura 50: Fiabilidad y estabilidad

Consumo de batería

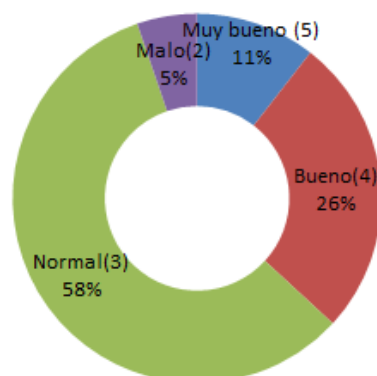


Figura 51: Encuesta Consumo de batería

Velocidad

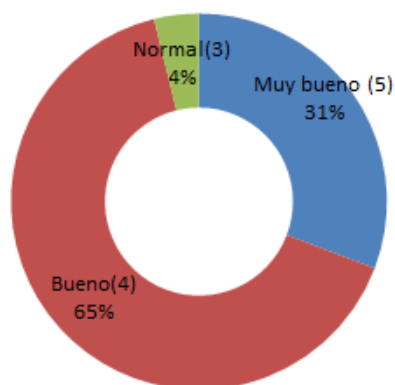


Figura 52: Encuesta Velocidad

Ordenación de los ejercicios

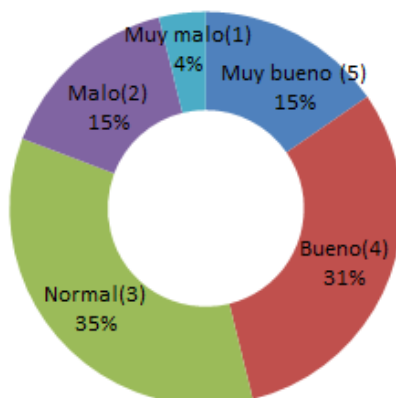


Figura 53: Encuesta Ordenación de los ejercicios

En general los resultados obtenidos han sido buenos, a excepción de la fiabilidad y estabilidad el resto de puntos de la encuesta obtienen una nota media superior a “Normal” y muy cercana o superior a “Bueno”. La realización de la encuesta ha servido para corregir algunos fallos, de ahí que la fiabilidad tuviese unos resultados más bajos de lo esperado, y para incluir algunas futuras mejoras que han sugerido los alumnos.

6.3 Trabajo futuro

Dado que la utilización de dispositivos móviles para fines educativos tiene muchas ventajas, la idea principal es que se continúe desarrollando la aplicación y mejorándola en futuros proyectos de fin de carrera.

A continuación se muestran algunos de los puntos más importantes para futuras mejoras:

- Posibilidad de hacer zoom en el tutorial y en las imágenes de los circuitos
- Cambiar la manera de volver a ver el circuito del enunciado para que sea más rápido y eficaz. (Deslizando el dedo como en el tutorial)
- Diseñar una manera de cronometrar el tiempo al resolver los ejercicios
- Cambiar la forma de indicar los fallos cometidos, por ejemplo: que las celdas erróneas se pongan en rojo
- Opción para marcar los ejercicios que ya se han realizado
- Añadir una pregunta al final del ejercicio para que el alumno además de resolver el problema indique de qué tipo de circuito se trata (And, Or...)
- La posibilidad de que el usuario dibuje las tablas en vez de dárselas hechas
- Incorporar un sistema de puntuación de los ejercicios
- Añadir ejercicios de síntesis de circuitos
- Añadir una hoja de anotaciones del usuario en cada ejercicio
- Adaptar a los principales sistemas operativos, (iOS y Windows Phone)
- Traducir a diferentes idiomas
- Añadir más preguntas al test
- Establecer una cuenta atrás en el test
- Modificar la interfaz de usuario

También se podrían estudiar mejoras más complejas como:

- Análisis y optimización del consumo de batería
- Optimización del código
- Utilizar un servidor para almacenar imágenes y disminuir así el peso de la aplicación
- Estudiar un mecanismo que haga posible diseñar nuevos ejercicios desde la aplicación y que sea posible compartirlos con otros usuarios
- Mensajería interna entre usuarios

Si quisiéramos ganar dinero con la aplicación deberíamos desarrollar alguna de las siguientes opciones:

- Convertirla en una aplicación de pago, a pesar de que obtendríamos ingresos por usuario, esta opción no es la adecuada ya que requeriría promocionar la aplicación y tendría menos alcance. Si eligiésemos esta opción el 30% de los ingresos serían para Google y el 70% para el creador.
- Añadir publicidad a la aplicación, los ingresos variarían en función del tipo de anuncio y del gestor de publicidad utilizado. Suponiendo que utilizamos AdMob como gestor de publicidad, ya que es uno de los más utilizados y escogemos anuncios poco invasivos, banners de texto o gráficos, los ingresos serían del orden de 40€ cada 500.000 impresiones. (Una impresión es cada vez que se muestra un anuncio en la pantalla del móvil) Aunque esta estimación es muy variable ya que existen múltiples opciones de pago, por impresión, por click, por acción... y varían según el idioma.



Figura 54: Ingresos publicidad/idioma

- Aplicación gratuita con módulos de pago, si la aplicación es buena será rentable. En nuestro caso podríamos añadir módulos con más ejercicios, diferentes tipos de exámenes o eliminar las soluciones de la app y crear un módulo de pago con las soluciones.

Tras realizar este pequeño estudio creemos que en futuras mejoras podríamos añadir publicidad a la aplicación o desarrollar módulos de pago.

Referencias

- [1] Francisco Javier Ceballos. “JAVA 2 Curso de programación. “
- [2] Francisco Javier Ceballos Sierra. “Curso de programación C/C++”
- [3] Jesús Tomás Girones. “El gran libro de Android, 2011”
- [4] EPS UAM “Desarrollo de aplicaciones para entornos móviles”
- [5] Roger Candenhead & Laura Lemay. “Teach Yourself Java 6 in 21 days”
- [6] <https://developer.android.com>
- [7] <http://stackoverflow.com>
- [8] <http://www.androidcurso.com>
- [9] <http://www.aprendeandroid.com/menu.htm>
- [10] <http://www.sgoliver.net/blog/?p=1731>
- [11] <http://es.wikipedia.org/wiki/Android>
- [12] <http://www.elandroidelibre.com>
- [13] <http://www.android-so.com>
- [14] <http://gs.statcounter.com>
- [15] <https://play.google.com/store/apps>
- [16] <http://androidzone.org/2013/05/historia-de-android-la-evolucion-a-lo-largo-de-sus-versiones/>
- [17] <https://sites.google.com/site/swcuc3m/home/android/generalidades/2-2-arquitectura-de-android>
- [18] <http://www.idc.com/prodserv/smartphone-os-market-share.jsp>
- [19] <http://www.emezeta.com/articulos/conferencia-ganar-dinero-con-android>
- [20] <https://androidasilearnit.wordpress.com/2011/03/05/custom-toggle-button/>

Glosario

ADT: Android Development Tools

API: Application Programming Interface

ARM: Advanced RISC Machine

AVD: Android Virtual Device

Driving: Cantidad de corriente que da una salida respecto al valor estándar.

DIE: Dispositivos Integrados Especializados

DsLab: Digital System Laboratory

EPS: Escuela Politécnica Superior

MOS: Metal-Oxide Semiconductor

Mux: Multiplexor

PDF: Portable Document Format

SDK: Software Development Kit

SO: Sistema Operativo

Streaming: Reproducción de archivos sin necesidad de descargarlos.

Tethering : Proceso por el cual un dispositivo móvil con conexión a Internet actúa como pasarela para ofrecer acceso a la red a otros dispositivos.

Tri-state: Tercer estado

UAM: Universidad Autónoma de Madrid

Anexos

A) PRESUPUESTO

1) Ejecución Material

- Compra de ordenador personal (Software incluido)..... 600 €
- Móvil Android.....300€
- Tableta Android250€
- Material de oficina10€
- Total de ejecución material 1.160€

2) Gastos generales

- 16 % sobre Ejecución Material 185,6€

3) Beneficio Industrial

- 6 % sobre Ejecución Material 69,6€

4) Honorarios Proyecto

- 540 horas a 15 € / hora 8.100€

5) Material fungible

- Gastos de impresión 55 €
- Encuadernación 20 €

6) Subtotal del presupuesto

- Subtotal Presupuesto 9.590,2€

7) I.V.A. aplicable

- 21% Subtotal Presupuesto 2.013,942 €

8) Total presupuesto

- Total Presupuesto 11.604, 142€

Madrid, Enero de 2015

El Ingeniero Jefe de Proyecto

Fdo.: Ana González Fernández

Ingeniero de Telecomunicación

PLIEGO DE CONDICIONES

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, de una APLICACIÓN ANDROID PARA LA ENSEÑANZA DE TEMAS DE CIRCUITOS INTEGRADOS BÁSICOS MOS. En lo que sigue, se supondrá que el proyecto ha sido encargado por una empresa cliente a una empresa consultora con la finalidad de realizar dicho sistema. Dicha empresa ha debido desarrollar una línea de investigación con objeto de elaborar el proyecto. Esta línea de investigación, junto con el posterior desarrollo de los programas está amparada por las condiciones particulares del siguiente pliego.

Supuesto que la utilización industrial de los métodos recogidos en el presente proyecto ha sido decidida por parte de la empresa cliente o de otras, la obra a realizar se regulará por las siguientes:

Condiciones generales

1. La modalidad de contratación será el concurso. La adjudicación se hará, por tanto, a la proposición más favorable sin atender exclusivamente al valor económico, dependiendo de las mayores garantías ofrecidas. La empresa que somete el proyecto a concurso se reserva el derecho a declararlo desierto.

2. El montaje y mecanización completa de los equipos que intervengan será realizado totalmente por la empresa licitadora.

3. En la oferta, se hará constar el precio total por el que se compromete a realizar la obra y el tanto por ciento de baja que supone este precio en relación con un importe límite si este se hubiera fijado.

4. La obra se realizará bajo la dirección técnica de un Ingeniero Superior de Telecomunicación, auxiliado por el número de Ingenieros Técnicos y Programadores que se estime preciso para el desarrollo de la misma.

5. Aparte del Ingeniero Director, el contratista tendrá derecho a contratar al resto del personal, pudiendo ceder esta prerrogativa a favor del Ingeniero Director, quien no estará obligado a aceptarla.

6. El contratista tiene derecho a sacar copias a su costa de los planos, pliego de condiciones y presupuestos. El Ingeniero autor del proyecto autorizará con su firma las copias solicitadas por el contratista después de confrontarlas.

7. Se abonará al contratista la obra que realmente ejecute con sujeción al proyecto que sirvió de base para la contratación, a las modificaciones autorizadas por la superioridad o a las órdenes que con arreglo a sus facultades le hayan comunicado por escrito al Ingeniero Director de obras siempre que dicha obra se haya ajustado a los preceptos de los pliegos de condiciones, con arreglo a los cuales, se harán las modificaciones y la valoración de las diversas unidades sin que el importe total pueda exceder de los presupuestos aprobados. Por consiguiente, el número de unidades que se consignan en el proyecto o en el presupuesto, no podrá servirle de fundamento para entablar reclamaciones de ninguna clase, salvo en los casos de rescisión.

8. Tanto en las certificaciones de obras como en la liquidación final, se abonarán los trabajos realizados por el contratista a los precios de ejecución material que figuran en el presupuesto para cada unidad de la obra.

9. Si excepcionalmente se hubiera ejecutado algún trabajo que no se ajustase a las condiciones de la contrata pero que sin embargo es admisible a juicio del Ingeniero Director de obras, se dará conocimiento a la Dirección, proponiendo a la vez la rebaja de precios que el Ingeniero estime justa y si la Dirección resolviera aceptar la obra, quedará el contratista obligado a conformarse con la rebaja acordada.

10. Cuando se juzgue necesario emplear materiales o ejecutar obras que no figuren en el presupuesto de la contrata, se evaluará su importe a los precios asignados a otras obras o materiales análogos si los hubiere y cuando no, se discutirán entre el Ingeniero Director y el contratista, sometiéndolos a la aprobación de la Dirección. Los nuevos precios convenidos por uno u otro procedimiento, se sujetarán siempre al establecido en el punto anterior.

11. Cuando el contratista, con autorización del Ingeniero Director de obras, emplee materiales de calidad más elevada o de mayores dimensiones de lo estipulado en el proyecto, o

sustituya una clase de fabricación por otra que tenga asignado mayor precio o ejecute con mayores dimensiones cualquier otra parte de las obras, o en general, introduzca en ellas cualquier modificación que sea beneficiosa a juicio del Ingeniero Director de obras, no tendrá derecho sin embargo, sino a lo que le correspondería si hubiera realizado la obra con estricta sujeción a lo proyectado y contratado.

12. Las cantidades calculadas para obras accesorias, aunque figuren por partida alzada en el presupuesto final (general), no serán abonadas sino a los precios de la contrata, según las condiciones de la misma y los proyectos particulares que para ellas se formen, o en su defecto, por lo que resulte de su medición final.

13. El contratista queda obligado a abonar al Ingeniero autor del proyecto y director de obras así como a los Ingenieros Técnicos, el importe de sus respectivos honorarios facultativos por formación del proyecto, dirección técnica y administración en su caso, con arreglo a las tarifas y honorarios vigentes.

14. Concluida la ejecución de la obra, será reconocida por el Ingeniero Director que a tal efecto designe la empresa.

15. La garantía definitiva será del 4% del presupuesto y la provisional del 2%.

16. La forma de pago será por certificaciones mensuales de la obra ejecutada, de acuerdo con los precios del presupuesto, deducida la baja si la hubiera.

17. La fecha de comienzo de las obras será a partir de los 15 días naturales del replanteo oficial de las mismas y la definitiva, al año de haber ejecutado la provisional, procediéndose si no existe reclamación alguna, a la reclamación de la fianza.

18. Si el contratista al efectuar el replanteo, observase algún error en el proyecto, deberá comunicarlo en el plazo de quince días al Ingeniero Director de obras, pues transcurrido ese plazo será responsable de la exactitud del proyecto.

19. El contratista está obligado a designar una persona responsable que se entenderá con el Ingeniero Director de obras, o con el delegado que éste designe, para todo relacionado con ella. Al

ser el Ingeniero Director de obras el que interpreta el proyecto, el contratista deberá consultarle cualquier duda que surja en su realización.

20. Durante la realización de la obra, se girarán visitas de inspección por personal facultativo de la empresa cliente, para hacer las comprobaciones que se crean oportunas. Es obligación del contratista, la conservación de la obra ya ejecutada hasta la recepción de la misma, por lo que el deterioro parcial o total de ella, aunque sea por agentes atmosféricos u otras causas, deberá ser reparado o reconstruido por su cuenta.

21. El contratista, deberá realizar la obra en el plazo mencionado a partir de la fecha del contrato, incurriendo en multa, por retraso de la ejecución siempre que éste no sea debido a causas de fuerza mayor. A la terminación de la obra, se hará una recepción provisional previo reconocimiento y examen por la dirección técnica, el depositario de efectos, el interventor y el jefe de servicio o un representante, estampando su conformidad el contratista.

22. Hecha la recepción provisional, se certificará al contratista el resto de la obra, reservándose la administración el importe de los gastos de conservación de la misma hasta su recepción definitiva y la fianza durante el tiempo señalado como plazo de garantía. La recepción definitiva se hará en las mismas condiciones que la provisional, extendiéndose el acta correspondiente. El Director Técnico propondrá a la Junta Económica la devolución de la fianza al contratista de acuerdo con las condiciones económicas legales establecidas.

23. Las tarifas para la determinación de honorarios, reguladas por orden de la Presidencia del Gobierno el 19 de Octubre de 1961, se aplicarán sobre el denominado en la actualidad “Presupuesto de Ejecución de Contrata” y anteriormente llamado “Presupuesto de Ejecución Material” que hoy designa otro concepto.

Condiciones particulares

La empresa consultora, que ha desarrollado el presente proyecto, lo entregará a la empresa cliente bajo las condiciones generales ya formuladas, debiendo añadirse las siguientes condiciones particulares:

1. La propiedad intelectual de los procesos descritos y analizados en el presente trabajo, pertenece por entero a la empresa consultora representada por el Ingeniero Director del Proyecto.

2. La empresa consultora se reserva el derecho a la utilización total o parcial de los resultados de la investigación realizada para desarrollar el siguiente proyecto, bien para su publicación o bien para su uso en trabajos o proyectos posteriores, para la misma empresa cliente o para otra.

3. Cualquier tipo de reproducción aparte de las reseñadas en las condiciones generales, bien sea para uso particular de la empresa cliente, o para cualquier otra aplicación, contará con autorización expresa y por escrito del Ingeniero Director del Proyecto, que actuará en representación de la empresa consultora.

4. En la autorización se ha de hacer constar la aplicación a que se destinan sus reproducciones así como su cantidad.

5. En todas las reproducciones se indicará su procedencia, explicitando el nombre del proyecto, nombre del Ingeniero Director y de la empresa consultora.

6. Si el proyecto pasa la etapa de desarrollo, cualquier modificación que se realice sobre él, deberá ser notificada al Ingeniero Director del Proyecto y a criterio de éste, la empresa consultora decidirá aceptar o no la modificación propuesta.

7. Si la modificación se acepta, la empresa consultora se hará responsable al mismo nivel que el proyecto inicial del que resulta el añadirla.

8. Si la modificación no es aceptada, por el contrario, la empresa consultora declinará toda responsabilidad que se derive de la aplicación o influencia de la misma.

9. Si la empresa cliente decide desarrollar industrialmente uno o varios productos en los que resulte parcial o totalmente aplicable el estudio de este proyecto, deberá comunicarlo a la empresa consultora.

10. La empresa consultora no se responsabiliza de los efectos laterales que se puedan producir en el momento en que se utilice la herramienta objeto del presente proyecto para la realización de otras aplicaciones.

11. La empresa consultora tendrá prioridad respecto a otras en la elaboración de los proyectos auxiliares que fuese necesario desarrollar para dicha aplicación industrial, siempre que no haga explícita renuncia a este hecho. En este caso, deberá autorizar expresamente los proyectos presentados por otros.

12. El Ingeniero Director del presente proyecto, será el responsable de la dirección de la aplicación industrial siempre que la empresa consultora lo estime oportuno. En caso contrario, la persona designada deberá contar con la autorización del mismo, quien delegará en él las responsabilidades que ostente.

